НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 «Програмна інженерія»

на тему: «Інтерактивна веб-дошка для зберігання особистого медіа-контенту»

Виконала:
студентка IV курсу, групи КП-51
Дячук Дар'я Сергіївна _____

Керівник:
Старший викладач кафедри ПЗКС,
Гадиняк Р.А. _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н.,
Онай М.В. _____

Рецензент:
Доцент кафедри ММСА, к.т.н., доцент,
Дідковська М.В. _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2019 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки – 6.050103 «Програмна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ І.А. Дичка

«___»_____2018 р.

ЗАВДАННЯ

на дипломний проект студенту

Дячук Дар'ї Сергіївні

1. Тема проекту «Інтерактивна веб-дошка для зберігання особистого медіа-контенту», керівник проекту Гадиняк Руслан Анатолійович, старший викладач кафедри ПЗКС, затверджені наказом по університету від «22» травня 2019 р. №1331-С

2. Термін подання студентом проекту «20» червня 2019 р.

3. Вихідні дані до проекту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - Огляд існуючих програмних рішень
   - Обґрунтування вибору засобів реалізації
   - Структурна організація розробленого програмного забезпечення
   - Аналіз розробленого веб-застосунку

5. Перелік обов'язкового графічного матеріалу:
   - Схема бази даних (креслення).
   - Діаграма використання (креслення).
   - Структура серверної та клієнтської частини веб-застосунку (плакат)
   - Діаграма діяльності (плакат).

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2018 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Вивчення літератури за тематикою проекту	17.11.2018	
2.	Розроблення та узгодження технічного завдання	28.11.2018	
3.	Аналіз існуючих рішень	15.12.2018	
4.	Підготовка матеріалів першого розділу дипломного проекту	30.12.2018	
5.	Розроблення дизайну сторінок та графічних елементів	03.02.2019	
6.	Підготовка матеріалів другого розділу дипломного проекту	20.02.2019	
7.	Програмна реалізація веб-застосунку	17.03.2019	
8.	Підготовка матеріалів третього розділу дипломного проекту	30.03.2019	
9.	Тестування веб-застосунку	05.04.2019	
10.	Підготовка матеріалів четвертого розділу дипломного проекту	11.04.2019	
11.	Підготовка графічної частини дипломного проекту	12.05.2019	
12.	Оформлення документації дипломного проекту	26.05.2019	

Студент Д.С. Дячук

Керівник проекту Р.А. Гадиняк

АНОТАЦІЯ

Даний дипломний проект присвячений створенню веб-застосунку для зберігання особистого унікального медіа-контенту та інтерактивної взаємодії з ним.

У роботі виконано порівняльний аналіз існуючих рішень для роботи з медіа-файлами, такими як зображення, текст і аудіо, доступних можливостей для взаємодії з ними, обґрунтовано вибір технологій та допоміжних бібліотек серверної та клієнтської частин для реалізації даного веб-застосунку. Розроблений веб-застосунок надає користувачам можливість завантажити на інтерактивну веб-дошку зображення, аудіо або текст у вигляді нотатки. У кожного типу медіа-файлу за замовчуванням є власне оформлення, яке користувач може змінити за бажанням. Доданий на дошку контент можна переміщати, повертати та змінювати у розмірах. Користувач може створити декілька дошок, зробити їх публічними (доступними для перегляду іншим) або приватними.

У даному дипломному проекті розроблено: архітектуру серверної та клієнтської частини веб-застосунку, програмні модулі для роботи з медіа-файлами (зображення, аудіо, текст) у середовищі canvas, а також графічні елементи та дизайн веб-сторінок.

ABSTRACT

This diploma project is devoted to creating a Web application for storing personalized and unique media content and interacting with it.

In this work, a comparative analysis of existing solutions for working with media files such as images, text and audio, available interoperability options has been made, the choice of technologies and auxiliary libraries for server and client parts for the implementation of this web application has been substantiated. The developed web application provides users with the ability to upload images, audio or text as a note to an interactive whiteboard. Each type of media file has its own default design, which the user can change at will. The contents of the board can be moved, rotated and resized. The user can create several boards, make them public (available for viewing by others) or private.

This graduation project has developed: the architecture of the server and client part of the web application, software modules for working with media files (image, audio, text) in the canvas environment, as well as graphic elements and web page design.

ДП.045440-01-90 Інтерактивна веб-дошка для зберігання особистого медіа-контенту. Відомість проекту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проекту		
ДП.045440-02-91	Інтерактивна веб-дошка	5	
	для зберігання особистого		
	медіа-контенту.		
	Технічне завдання		
ДП.045440-03-81	Інтерактивна веб-дошка	6	
	для зберігання особистого		
	медіа-контенту.		
	Пояснювальна записка		
ДП.045440-04-51	Інтерактивна веб-дошка	4	
	для зберігання особистого		
	медіа-контенту.		
	Програма та методика		
	тестування		
ДП.045440-05-34	Інтерактивна веб-дошка	1	
	для зберігання особистого		
	медіа-контенту.		
	Керівництво користувача		
ДП.045440-06-99	Інтерактивна веб-дошка	1	
	для зберігання особистого		
	медіа-контенту.		
	Функціональні		
	можливості		
	веб-застосунку. Діаграма		
	використання		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-07-99	Інтерактивна веб-дошка	1	
	для зберігання особистого		
	медіа-контенту.		
	Архітектура		
	веб-застосунку. Схема		
	бази даних		
ДП.045440-08-98	Інтерактивна веб-дошка	1	
	для зберігання особистого		
	медіа-контенту.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

ІНТЕРАКТИВНА ВЕБ-ДОШКА ДЛЯ ЗБЕРІГАННЯ ОСОБИСТОГО МЕДІА-КОНТЕНТУ

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проекту:

_____ Р.А. Гадиняк

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ Д.С. Дячук

2018

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Інтерактивна веб-дошка для зберігання особистого медіа-контенту.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання в якості веб-застосунку, який дозволить завантажити особистий медіа-контент та забезпечить інтерактивну взаємодію з ним. Мета веб-застосунку полягає у зберіганні особистих важливих, пам'ятних творчих надбань та розвитку творчих здібностей завдяки інтерактивним можливостям.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Веб-застосунок повинен забезпечувати такі основні функції:

- Створити власний профіль.
- Можливість створити декілька дошок.
- Видалити дошку.
- Додати на дошку текст.
- Додати на дошку зображення.

- Додати на дошку аудіо.
- Можливість відфільтрувати контент на дошці за типом.
- Змінити розмір та кут повороту об’єкта на дошці.
- Розмістити об’єкт на дошці у будь-якому місці.
- Зберегти стан дошки.
- Переглянути завантажене зображення у повному розмірі.
- Зробити дошку публічною або приватною, тобто доступною для перегляду іншим користувачам.
- Динамічне оновлення сторінки без перезавантаження.

Додаткові вимоги:

- Наявність анімованих кнопок.
- Дизайн сторінок з використанням трьох акцентних кольорів: зелений, рожевий, помаранчевий.
- Всі іконки веб-застосунку мають бути в одному стилі.

5. ВИМОГИ ДО ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

1) пояснювальна записка;
2) програма та методика тестування;
3) керівництво користувача;
4) креслення:
   - «Інтерактивна веб-дошка для зберігання особистого медіа-контенту. Архітектура веб-застосунку. Схема бази даних»;
   - «Інтерактивна веб-дошка для зберігання особистого медіа-контенту. Функціональні можливості веб-застосунку. Діаграма використання».

6. ЕТАПИ ПРОЕКТУВАННЯ

Вивчення літератури за тематикою проекту..……..14.11.2018

Розроблення та узгодження технічного завдання……....28.11.2018

Розроблення структури веб-застосунку..…….....20.12.2018

Розроблення дизайну сторінок та графічних елементів……..03.02.2019

Програмна реалізація веб-застосунку...……....17.03.2019

Тестування веб-застосунку…….....05.04.2019

Підготовка матеріалів текстової частини проекту……..….28.04.2019

Підготовка матеріалів графічної частини проекту……...…12.05.2019

Оформлення технічної документації проекту……..…25.05.2019

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ІНТЕРАКТИВНА ВЕБ-ДОШКА ДЛЯ ЗБЕРІГАННЯ ОСОБИСТОГО МЕДІА-КОНТЕНТУ

Пояснювальна записка

ДП.045440-03-81

"ПОГОДЖЕНО"

Керівник проекту:

_____ Р.А. Гадиняк

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ Д.С. Дячук

2019

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ І ПОЗНАЧЕНЬ

Веб-застосунок – розподілений застосунок, в якому сервером виступає веб-сервер, а клієнтом – браузер. Логіка застосунку реалізовується на сервері, а браузер переважно показує інформацію, завантажену мережею з сервера, і передає назад дані користувача.

Фреймворк (англ. framework) – каркас програмного забезпечення, що полегшує процес розроблення завдяки готовим компонентам та базовим модулям.

Бібліотека – збірка підпрограм чи об'єктів, які використовуються для розроблення програмного забезпечення.

CSS-препроцесор – програма, що має власний синтаксис, але може згенерувати з нього CSS-код.

UML (англ. Unified Modeling Language) – уніфікована мова моделювання, використовується у парадигмі об’єктно-орієнтованого програмування. Є невід’ємною частиною уніфікованого процесу розроблення програмного забезпечення.

NPM – менеджер пакетів для JavaScript і найбільший реєстр програмного забезпечення на платформі Node.js.

Веб-сервер – сервер, що приймає HTTP-запити від клієнтів, зазвичай веб-браузерів, і повертає HTTP-відповіді, як правило, разом з HTML-сторінкою, зображенням, файлом або іншими даними.

Хмарне сховище – модель онлайн-сховища, в якому дані зберігаються на численних розподілених в мережі серверах, які надаються клієнтам на використання. Дані зберігаються й обробляються на так званій «хмарі», яка з точки зору клієнта представляє собою один великий віртуальний сервер.

Маршрутизація – визначає як застосунок відповідає на запит клієнта за конкретною адресою (кінцевій точці), якою є URI (або шлях), і певному методу запиту HTTP.

Рендеринг (англ. rendering – «візуалізація») – термін в комп’ютерній графіці, який означає процес отримання зображення моделі за допомогою комп’ютерної програми, де модель – це опис будь-яких об’єктів або явищ на строго визначеній мові або у вигляді структури даних.

CSS (англ. Cascading Style Sheets) – каскадна таблиця стилів.

Bucket – окремий кошик у хмарному сховищі AWS S3, де зберігаються файли.

Drag-and-Drop (перетягування) – спосіб керування елементами інтерфейсу за допомогою миші або сенсорного екрану.

DOM (англ. Document Object Model) – об’єктна модель документа.

JSON (англ. JavaScript Object Notation) – Текстовий формат обміну даними, оснований на JavaScript.

HTML (англ. Hypertext Markup Language) – мова розмітки гіпертексту.

HTTP (англ. Hypertext Transfer Protocol) – протокол передачі гіпертексту.

SVG (англ. Scalable Vector Graphics) – масштабована векторна графіка.

URI (англ. англ. Uniform Resource Identifier) – уніфікований ідентифікатор ресурсу.

СУБД – система управління базами даних.

ВСТУП

У сучасному світі завдяки розвинутим новітнім технологіям ми маємо широкі можливості для прояву творчості кардинально різного характеру і збереження результатів творчого процесу у різних форматах. Наприклад, аудіо- та відеофайли, цифрові зображення, текстові дані. Раніше письменники були обмежені кількістю паперу та чорнил, художники – витратними матеріалами на кшталт фарб, полотна, пензликів тощо, а фільми знімалися на плівку, і переважна більшість людей не мала змоги вільно втілити власні ідеї у життя. У XXI сторіччі майже кожен має смартфон, персональний комп'ютер або фотокамеру, розроблено багато програмного забезпечення для роботи з медіа-контентом. Тому зовсім не обов'язково купувати фарби, папір, музичні інструменти або окрему камеру, щоб зробити свою фантазію реальністю.

Безсумнівною перевагою вищезгаданих апаратно-програмних засобів є те, що вони зазвичай мають великий обсяг пам'яті, завдяки чому у нас накопичується велика кількість медіа-файлів. Дана проблема особливо знайома людям, що захоплюються різноманітними хобі. Тому виникає питання: де саме тримати свої творчі надбання?

Наразі існує багато тематичних ресурсів для зберігання медіа-файлів, однак вони переважно спрямовані на роботу з одним типом контенту. Звичайно, існують ресурси, що дозволяюсь зберігати всі типи медіа-файлів, однак їх проблема полягає в обмеженій кількості дій, слідуванні певним, строго визначеним розробниками командам і відсутності підтримки творчого самовираження та механізму підкреслення важливості деяких творінь, адже іноді вкрай необхідним є збереження пам'ятних речей, які можуть загубитися серед потоку буденних дрібниць.

Отже, актуальною постає задача розроблення програмного забезпечення, завдяки якому користувач зможе зберігати особисті важливі, пам'ятні творчі надбання у вигляді медіа-контенту і характерною рисою якого

є організація інтерактивної взаємодії з кожним файлом, призначення йому оформлення за бажанням.

1. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

1.1. Обґрунтування вибору критеріїв оцінювання

Програми з широкими можливостями взаємодії користувача з інтерфейсом завжди привертали більше уваги та викликали зацікавленість, особливо коли доступна кількість дій максимально необмежена.

Головною метою даного проекту є збереження особистого пам’ятного медіа-контенту. Завдяки сучасним технологічним можливостям багато людей стикається з проблемою великого накопичення медіа-файлів, тому можна точно зазначити, що серед всіх своїх творінь людина може виділити лише декілька особливо важливих, пам’ятних або надзвичайно вдалих. Це може бути емоційний вірш, перша фотографія дитини або найкращий малюнок за все життя. І такі речі губляться серед буденних дрібниць, а іноді навіть випадково видаляються. В реальному житті так само існує вірогідність викинути важливе творіння з непотрібним мотлохом або, наприклад, при переїзді до іншого місця проживання.

Щоб робота з веб-застосунком була максимально зручною, необхідно забезпечити інтерактивність під час заповнення дошки відповідним контентом.

Для успішної реалізації проекту необхідно визначитися з платформою, на якій він буде працювати. На сьогодні, найбільш популярними рішеннями є розроблення веб-застосунку та мобільного додатку. Згідно з дослідженнями [1], Інтернет-користувачі надають перевагу браузерам для шопінгу, пошуку інформації та розваг, в той час як мобільні додатки використовуються, якщо необхідно зберігати дані або спілкуватися з іншими людьми.

Проаналізуємо обидва варіанти.

1.1.1. *Мобільний додаток*

Зараз одним з найбільш популярних пристроїв у світі вважається смартфон, основними перевагами якого є портативність, велика кількість різноманітних функцій та безпосередньо сенсорний екран. Він значно

розширює користувацькі можливості і робить взаємодію з телефоном простішою, зрозумілою. Сенсорний екран надає більше можливостей для інтерактивної взаємодії з інтерфейсом програми, зокрема завдяки жестам, які використовувати набагато зручніше, аніж постійно натискати клавішу миші.

Мобільний додаток надсилає сповіщення. Вони нагадують користувачу про програму, що дуже корисно, особливо, якщо на смартфоні встановлено багато додатків. Сповіщення спонукають робити дію, а, отже, забезпечують більш активне використання програмного забезпечення.

У той самий час, смартфони переважно мають невеликий розмір екрану. Тому не всі програми підходять під реалізацію мобільного додатку з таких причин:

- Незручна взаємодія з об’єктами. Якщо об’єктів на екрані багато або вони малого розміру, то складно обрати для роботи бажаний. Модифікація у даному випадку є ще більш великою проблемою, особливо, якщо у програмі не передбачене масштабування.
- Проблема масштабування. Маленькі екрани смартфонів потребують продуманого інтерфейсу. В даному випадку, постійне збільшення дошки для розгляду конкретного медіа-контенту буде недоречним. Користувач повинен бачити загальний вигляд у зручному для нього масштабі, що реалізувати на екрані смартфона неможливо через великий розмір дошки. Тому контент буде занадто малого розміру.
- Проблема жестів. Більшість смартфонів підтримують різноманітні функції жестів, однак при хибному русі пальців, смартфон може зреагувати неправильно, що часто призводить до помилок в роботі та виконанні небажаних команд.

Здавалося б проблему малого екрану можуть вирішити планшети, адже вони набагато більшого розміру. Однак дослідження [2] показують, що попит на планшети кожного року все більше знижується. Сьогоднішніх розмірів телефона користувачам вистачає для вирішення майже всіх задач. В іншому випадку вони використовують персональний комп’ютер або ноутбук.

Мобільний додаток у будь-якому випадку займає пам’ять смартфона, навіть якщо усі дані введені або завантажені користувачем зберігаються у хмарному сховищі. Будь-який зайвий додаток може знижувати загальну продуктивність телефона, особливо якщо завантажені дані зберігаються безпосередньо на пристрої.

1.1.2. *Веб-застосунок*

Одна з головних переваг веб-застосунку – це доступ до нього з великої кількості пристроїв: комп’ютера, ноутбука, телефона тощо. Адже для роботи з програмою користувачу лише необхідні інтернет-підключення та браузер, який зазвичай не займає багато пам’яті.

Усі персональні та завантажені дані веб-застосунку зберігаються у хмарному сховищі, що дає змогу користувачу зайти у свій профіль з будь-якого браузера та пристрою і продовжити роботу з того моменту, на якому він востаннє зупинився.

Веб-застосунками зазвичай користуються з комп’ютерів або ноутбуків, тому, завдяки великим екранам, можна реалізувати зручну взаємодію з об’єктами інтерфейсу, їх масштабування. Керування з комп’ютера більш точне, стабільне. Зокрема елементи управління, такі які мишка чи клавіатура, під’єднанні окремо, тому користувацький інтерфейс не перекривають додаткові спливаючі панелі, як у смартфонах. Більшість людей творчих професій, таких як програмісти, дизайнери, копірайтери працюють саме за великими екранами.

Отже, згідно проведеного аналізу, для даного проекту доцільно розробляти веб-застосунок, бо він буде відповідати таким критеріям:

- економія пам’яті пристрою;
- доступ з будь-якого пристрою, що підтримує інтернет-підключення;
- орієнтованість на великі екрани;
- забезпечення інтерактивності;
- зручне масштабування.

1.1.3. *Критерії оцінювання програмних засобів*

Було проведено дослідження, зокрема опитування зацікавлених сторін веб-застосунку, для визначення мінімально необхідного набору вимог до розроблюваного програмного забезпечення. Він зводиться до такого списку:

1. Можливість завантажити у свій профіль мультимедійний контент різного типу, а саме: текстові файли, зображення й аудіо.
2. Можливість завантаження медіа-контенту з веб-застосунку локально на пристрій.
3. Прослуховування завантажених аудіо-файлів.
4. Перегляд зображень у повному розмірі, адже під час загального перегляду не видно всіх деталей.
5. Забезпечення інтерактивності, тобто надати користувачу змогу розміщувати контент у будь-якому місці, змінювати його розміри та зовнішній вигляд. Дані функції мають бути доступними завжди.
6. Фільтрація розміщеного контенту за певними критеріями (наприклад, дата, тип) є важливою та зручною функцією, особливо коли у профілі велика кількість різноманітних об’єктів.
7. Зробити свою дошку публічною або приватною. Залежно від власних вподобань та цілей використання програмного забезпечення користувач повинен мати вибір надавати доступ іншим користувачам переглядати його завантажені файли чи ні.

1.2. Аналіз існуючих рішень

Для аналізу було обрано п’ять відомих програмних засобів, що забезпечують роботу з медіа-файлами та розраховані на використання творчими людьми.

1.2.1. *Behance*

Behance — це мережа сайтів і сервісів, що спеціалізуються на саморекламі, включаючи консалтинг та онлайн-портфоліо сайтів [3].

Спрямована на графічних дизайнерів, веб-дизайнерів, художників, ілюстраторів і навіть фотографів.

На даному ресурсі розміщують роботи переважно професіонали з метою просунути свої проекти або отримати роботу. Тому він не підходить для звичайних людей, для яких малювання чи дизайн є просто хобі.

Behance не дозволяє завантажувати аудіо-файли. Також тут немає можливості приховувати профіль від інших користувачів, так як це публічний ресурс, на якому люди реєструються у першу чергу заради самореклами, а не збереження особистих медіа-файлів.

Behance надає широкий список функцій для роботи з файлами (переважно зображеннями), зокрема їх редагування. Однак на ресурсі відсутня зазначена у вимогах інтерактивність, тобто завантажені файли показуються лише у вигляді стандартної сітки. Їх можна пересувати мишею, міняти місцями, але не можна змінити розмір та застосувати фільтрацію завантаженого контенту.

1.2.2. *Pinterest*

Pinterest – соціальний фото-сервіс для пошуку нових ідей [4]. Він дозволяє користувачам додавати зображення в режимі онлайн та розміщувати їх у тематичні колекції, і ділитися ними з іншими. Збережені зображення називаються пінами, а колекції – дошками.

Сервіс використовують як професіонали в сфері мистецтва, так і звичайні користувачі з метою простого створення власних колекцій шляхом додавання чужих робіт, які їм сподобались. Компанії також створюють власні піни, щоб збільшити об’єм продажів та підвищити рівень популярності бренду.

Pinterest дозволяє створити декілька дошок та налаштувати обмеження на перегляд. Дошки можуть бути публічними, тобто доступні всім користувачам ресурсу, або секретними – не доступні нікому, окрім власника та людей, яких він додав у список запрошених.

Усі зображення можна переглянути у повному розмірі, відкрити їх, щоб роздивитися детальніше. Також, є змога поміняти місцями елементи на дошці, просто перетягнувши мишею потрібний елемент на нове місце. Однак, доступні позиції зображень строго визначені: у форматі сітки, як на більшості ресурсів. Зберегти піни локально на комп'ютер не можна, але можна поділитися ними у соціальних мережах.

Pinterest дозволяє працювати тільки з зображеннями, завантажувати їх, описувати, коментувати. Динамічно змінювати їх розмір на дошці неможливо. Взаємодія з іншими форматами даних не передбачена, як і фільтрація файлів.

1.2.3. *Instagram*

Instagram – соціальна мережа для обміну та створення фотографій та відео з можливістю накладання фільтрів [5]. Його початковою головною особливістю було створення тільки квадратних фотографій, як у камерах моментального фотознімку.

Дехто використовує Instagram у якості власного блогу. Користувач може зробити профіль як публічним, так і приватним. Приватні профілі доступні тільки підписникам, яким власник дозволив спостерігати за своєю сторінкою.

Розміщені у соціальній мережі фотографії не можна завантажити на локальний пристрій. Усі пости розміщуються лише у табличному вигляді, тому у користувача не має змоги власноруч змінити зовнішній вигляд профілю. Медіа-контент можна завантажити лише двох типів: фото та відео.

Переважна кількість людей використовує Instagram для зображення повсякденного життя, буденних фотографій, серед яких особливі файли можуть дуже швидко губитися. Також, тут відсутня будь-яка фільтрація, сортування або пошук постів, тому, якщо у профілі користувача більше тисячі публікацій, необхідно витратити багато часу, щоб потрапити на початок.

Кількість функцій, можливостей, користувачів Instagram швидко зростає, що призводить до збільшення обсягу інформації, яка потрапляє на очі користувачам. Тому, виділити щось справді особливе і важливе стає складніше з кожним днем.

1.2.4. *Tumblr*

Tumblr – сервіс мікроблогів і соціальна мережа, що дозволяє публікувати зображення, відео, текстові повідомлення, аудіо-файли у своєму блозі [6].

Блогери мають змогу зробити свої блоги приватними або публічними. Tumblr надає широкі функціональні можливості завдяки роботі з медіа-файлами різного типу, які можна відкривати у повному розмірі, програвати, але не можна завантажити на локальний пристрій.

Пости у соціальній мережі мають подання у вигляді стрічки або таблиці, мають фіксований розмір. Змінити порядок публікацій або виконати фільтрацію не можна, адже їх за замовчуванням сортовано за датою.

Дизайн Tumblr простий, інтуїтивно зрозумілий. Свій профіль можна зробити унікальним в налаштуваннях, що дозволить змінити фон блогу, кольорову гаму та шрифти.

У даній соціальній мережі люди об’єднуються у групи за інтересами, діляться своїми думками з іншими, ведуть блоги. Для організації особистого, пам’ятного контенту Tumblr не є зручним, адже формат розміщення публікацій не достатньо гнучкий, можливості сортування та фільтрації відсутні, а взаємодії між користувачем і інтерфейсом програми не вистачає інтерактивності.

1.2.5. *Google Jamboard*

Jamboard – це інтерактивна дошка від компанії Google, оснащена сенсорним п’ятдесяти п’яти дюймовим дисплеєм [7]. Її основне призначення – ведення бізнес-проектів і навчальних процесів. Користувач починає свою роботу зі створення Jam-файлу, який представляє собою полотно для створення й редагування проектів.

Проте для роботи з Jam-файлами не обов’язково мати саму фізичну дошку, адже Jamboard інтегрується з багатьма іншими сервісами Google. Таким чином, отримати доступ до файлів з Google Drive [8], завантажити їх та

відредагувати свою віртуальну дошку можливо зі звичайного комп’ютера, ноутбука чи телефона.

На робоче полотно користувач має змогу додати нотаток або зображення, а завдяки інструменту перо намалювати будь-який малюнок. У версії Jamboard для персональних комп’ютерів не передбачено завантаження аудіо та відео на дошку. Усі об’єкти можна крутити, змінювати їх розмір та переставляти у будь-яке місце.

Jamboard має простий, зрозумілий інтерфейс та достатньо строгий дизайн. Неможливо обрати оформлення завантажених файлів, розробники надають доступ тільки до зміни кольору нотатка.

На дошку можна додати тільки ті зображення, які завантажені на Google Drive того ж облікового запису, однак не ті, що знаходяться локально на комп’ютері. Зображення на дошці неможливо відкрити у повному розмірі, тобто окремо на повний екран. Це може створювати проблеми при перегляді великих рисунків чи схем, тому що розмиваються дрібні деталі.

1.3. Результати аналізу

Виходячи з результатів проведеного аналізу, жодне з існуючих рішень не задовольняє в повній мірі сформульованому переліку вимог до розроблюваного програмного забезпечення. Відповідність існуючих програмних рішень поставленим вимогам можна наглядно побачити в табл. 1.

Загалом, усі наведені програми мають своє власне призначення, що майже повністю відрізняється від мети даного проекту – збереження особистого пам’ятного медіа-контенту. Instagram і Tumblr – соціальні мережі, де люди переважно діляться своїми думками, ведуть блоги на конкретну тему та слідкують один за одним. На таких ресурсах завжди багато зайвої інформації, велика кількість постів на сторінці профілю, що не дає змоги виділити по-справжньому важливі і особливі файли для користувача. На відміну від Instagram, Tumblr відповідає більшій кількості висунутим вимогам,

зокрема він підтримує взаємодію з різними типами файлів, а не тільки працює з зображеннями.

Behance – ресурс для професійних дизайнерів, які просувають свої роботи заради майбутнього працевлаштування. Він зовсім не підходить для звичайних користувачів.

Pinterest задовольняє деяким головним вимогам. Він один з небагатьох, який дозволяє частково інтерактивно взаємодіяти з елементами інтерфейсу, однак він спрямований на роботу лише з зображеннями, не має функції фільтрації контенту, а також його збереження на локальний пристрій.

Jamboard на перший погляд найкраще задовольняє поставленим вимогам, однак вона має досить малу кількість функціональних можливостей. Зокрема серед медіа-файлів можна завантажити лише зображення з Google Drive, а відсутність фільтрації контенту може стати значним мінусом, якщо на полотні розміщено багато об’єктів різного типу, що може викликати плутанину.

Головною проблемою усіх зазначених програмних рішень, окрім Jamboard, є відсутність інтерактивності, тобто неможливість змінювати розмір та дизайн завантаженого контенту, розміщувати його у будь-якій точці спеціально виділеного простору. В даних програмах контент упорядковується за часом публікації, тому користувачу необхідно довго гортати сторінку у пошуках старої публікації, що загубилась серед менш важливих. Також, лише два ресурси з п’яти надають змогу працювати не тільки з зображеннями, а й з текстовими і аудіо-файлами.

Таблиця 1

Відповідність існуючих рішень до поставлених вимог

	1	2	3	4	5	6	7
Behance				+	+/–		+
Pinterest				+	+/–		+
Instagram				+			+
Tumblr	+		+	+			+
Jamboard	+/–				+		+

Отже, на даний момент не існує відповідного програмного рішення для збереження особистого пам’ятного медіа-контенту. Усі згадані вище ресурси не задовольняють навіть половині поставлених вимог. Тому буде проведено розроблення програмного забезпечення, а саме веб-застосунку, який врахує усі висунуті вимоги.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Вибір мови програмування для розроблення серверної частини веб-застосунку

Згідно з постановкою задачі, вимогами до мови програмування є:

- наявність великої стандартної бібліотеки та фреймворків, що використовуються для розроблення серверної частини;
- висока продуктивність та швидкодія, зважаючи на значний обсяг переданих даних;
- забезпечення асинхронності;
- надання можливості обмінюватись даними у реальному часі.

2.1.1. *Python*

Python – високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробника [9]. Стандартна бібліотека включає великий обсяг корисних функцій.

Python підтримує кілька парадигм програмування. Основні архітектурні риси – динамічна типізація, автоматичне керування пам'яттю, механізм обробки виключень, підтримка багатопоточних обчислень і зручні високорівневі структури даних.

На відміну від багатьох мов програмування (C#, Java і т.д.) Python не вимагає опису змінних. Вони створюються на місці їх ініціалізації, а тому і тип змінної визначається в залежності від типу присвоєного значення.

Присутні засоби для роботи з багатьма мережевими протоколами і форматами Інтернету, наприклад, модулі для написання HTTP-серверів і клієнтів, для розбору і створення поштових повідомлень, для роботи з XML тощо. Набір модулів для роботи з операційною системою дозволяє писати кросплатформні додатки. Існують модулі для роботи з регулярними виразами, текстовим кодуванням, мультимедійними форматами, криптографічними протоколами, архівами, серіалізації даних, підтримка unit-тестування.

Однак, у Python наявні не менш важливі мінуси.

Зокрема, він має достатньо малу продуктивність та швидкодію, порівняно з більшістю інших мов програмувань таких як JavaScript, PHP, Java, Go. Це тому, що Python є інтерпретованою, а не компільованою мовою програмування. Окрім того, динамічна типізація спричиняє велику кількість помилок, більшість з яких показується лише під час виконання програми.

Під час розроблення веб-застосунків важливо пам’ятати про асинхронність – процес обробки введення/виводу, що дозволяє продовжити оброблення інших завдань, не чекаючи завершення попереднього завдання. Python не є ідеальним варіантом для асинхронного програмування.

2.1.2. *Ruby*

Ruby – це інтерпретована, повністю об'єктно-орієнтована мова програмування з чіткою динамічною типізацією [10]. Вона відома простим і чистим синтаксисом, схожим на синтаксис інших мов, таких як C++ і Perl. Має достатньо велику кількість бібліотек, готових для використання. Зокрема, це бібліотеки для роботи з різними мережевими протоколами на стороні сервера та клієнта, архівами, матрицями, засоби для системного адміністрування тощо. Також, мова надає можливість для роботи з різними форматами даних, відлагодження, володіє бібліотеками для unit-тестування. Ruby надає достатньо високу швидкодію, однак мова дуже вибаглива до ресурсів.

Найбільш популярними фреймворками є Rails, Rack, Sinatra, Hanami.

2.1.3. *JavaScript*

JavaScript – динамічна, об'єктно-орієнтована прототипна мова програмування [11]. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на клієнтській стороні взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки.

Як і Python, має динамічну типізацію. JavaScript підтримує не тільки прототипну парадигму проектування, а імперативну і часткового функціональну. Варто зазначити про деякі архітектурні властивості, а саме:

автоматичне керування пам’яттю, прототипне наслідування, функції як об’єкти першого класу.

Можна виділити такі основні переваги та можливості даної мови:

- жоден сучасний браузер не обходиться без підтримки JavaScript;
- JavaScript можна широко використовувати як на клієнтській, так і на серверній стороні;
- широкий вибір функціональних налаштувань;
- пряме підключення скриптів до HTML-коду;
- програми можна запускати і у браузері, і на сервері;
- робота з діями користувачів (кліки, переміщення курсора тощо);
- зміна стилів елементів.

На відміну від Python, JavaScript має набагато більшу швидкодію. Велика кількість фреймворків дозволяє розроблювати не тільки повноцінні сайти, а й функціональні модулі (різноманітні онлайн-інструменти), динамічні мобільні/настільні/веб додатки. Окрім фреймворків існує чимало бібліотек, які роблять процес написання коду набагато швидшим і простішим.

Для розроблювання сучасних веб-застосунків мовою JavaScript використовується платформа Node.js [12]. Вона використовується переважно на сервері, виконуючи роль веб-серверу.

Платформа побудована на розробленому компанією Google рушієві V8, що робить її однією з найбільш швидких у виконанні коду.

Для забезпечення оброблення великої кількості паралельних запитів у Node.js використовується асинхронна модель запуску коду, заснована на обробці подій в неблокуючому режимі та визначенні обробників зворотних викликів.

Для керування модулями використовується пакетний менеджер NPM (Node Package Manager), що включає в себе онлайн-базу даних публічних та приватних пакунків, яка називається реєстром. NPM складається з клієнта командного рядка, що взаємодіє з віддаленим реєстром. Це дозволяє

користувачам завантажувати та користуватися готовими модулями JavaScript і розповсюджувати їх.

Версії Node.js часто обновлюються, що може свідчити про достатню гарантію стабільності роботи.

Для Node.js існує велика кількість різних фреймворків. Одними з найпопулярніших є Express, Meteor, Koa, Sails, Mean.

Для розроблення серверної частини веб-застосунку доцільно обрати Node.js, адже він найкраще задовольняє усім вище поставленим вимогам. Дана платформа має дуже велику підтримку серед розробників, постійно оновлюється, має високу продуктивність. На відміну від Python та Ruby Node.js пропонує найкращу модель асинхронного програмування.

2.2. Вибір технологій для розроблення серверної частини

Веб-фреймворк полегшує процес розроблення складних веб-застосунків, позбавляючи від написання рутинного коду та зменшуючи вірогідність його дублювання. Зокрема розробнику не потрібно приділяти багато уваги низькорівневим деталям, таким як управління потоками чи процесами, сокетам тощо. Окрім того, фреймворки одразу надають базову стандартну структуру веб-застосунку, що дозволяє не витрачати багато часу на продумування та створення зручної ієрархії файлів.

Фреймворк для серверної частини веб-застосунку повинен відповідати таким вимогам як:

- сумісність з іншими інструментами, що планується використовувати (база даних, фреймворк для клієнтської частини тощо);
- наявність якісної документації;
- підтримка REST-архітектури;
- наявність спільноти, яка підтримує та випускає актуальні сумісні версії фреймворку.

Оскільки на сьогодні існує дуже велика кількість доступних для розроблення фреймворків, і більшість з них відповідають поставленим вище

вимогам, то для даного дипломного проекту було розглянуто та проаналізовано три найбільш популярних рішення.

2.2.1. *Express*

Express – мінімалістичний та гнучкий фреймворк для розроблення застосунків на Node.js [13]. Використовуваний розробниками давно, він є одним з найбільш популярних та стабільних рішень. Для цього фреймворку існує дуже багато детальних інструкцій, створених людьми, які перевірили його ефективність на практиці.

Головною особливістю Express є невеликий об’єм базового набору функціональних можливостей, а усі інші необхідні функції потрібно добирати за допомогою зовнішніх модулів. Тобто Express у чистому вигляді – це сервер, в якого може бути ні одного модуля. Завдяки цьому розробник з самого початку отримує легкий та швидкий інструмент, який він може розвивати та розширювати на свій розсуд. Фреймворк дозволяє вирішувати будь-які задачі, не обмежуючи розробника у виборі засобів.

Можна виділити такі основні плюси Express:

- простота;
- гнучкість;
- гарна масштабованість;
- велика розвинута спільнота;
- детальна документація;
- широкий вибір додаткових модулів.

2.2.2. *Koa.js*

Koa був створений як версія Express нового покоління, покращена, для створення веб-застосунків з високою продуктивністю [14]. Відповідно до цього, розробники намагались врахувати всі недоліки його попередника та зробити його більш сучасним й зручним у використанні.

Koa володіє майже такими самими функціональними можливостями, як і Express. На відміну він нього, фреймворк використовує не зворотні виклики, а генератори – тип функцій, які можуть бути запущені, зупинені і відновлені

незалежно від того, на якому етапі виконання вони знаходяться. Це дозволяє зменшити об’єм роботи з кодом та знижує вірогідність помилок.

Однак, зважаючи на відносно молодий вік, Koa не користується дуже широкою популярністю, тому не має великої підтримки спільноти, а також має трохи менше функціональних можливостей, на відміну від свого попередника.

2.2.3. *Sails.js*

Sails.js переважно створений для розроблення бізнес-застосунків та застосунків, що реалізують певні особливі функціональні можливості [15]. Загалом, Sails добре підходить для розроблення браузерних веб-застосунків, так як він добре працює з різними бібліотеками для клієнтської частини. Даний фреймворк розроблений як повністю готовий продукт, який містить достатньо функцій, щоб можна було почати роботу. Саме тому він від самого початку є досить важким та громіздким, що безумовно негативно впливає на швидкодію розроблюваного застосунку.

З одного боку, розробник може не сильно вдаватися у тонкощі процесу – можна просто взяти готове перевірене рішення. З іншого, розроблення застосунку буде обмежено наявними можливостями фреймворку, так як зовнішніх модулів для Sails набагато менше, ніж для Express або Koa.

Одною з головних особливостей даного фреймворку є вбудована технологія програмування Waterline ORM [16], яка використовується для забезпечення зв’язку з різними базами даних. Однак, не дивлячись на переваги цього компонента, у процесі розроблення можна стикнутися з труднощами. Наприклад, Waterline не підтримує транзакції, а оновлення та виправлення помилок відбуваються невчасно.

Отже, зважаючи на всі переваги й недоліки вище описаних фреймворків, для розроблення веб-застосунку було обрано Express, як найбільш гнучкий, швидкий, стабільний й перевірений. Він задовольняє усім вище описаним вимогам та дозволяє створити унікальне програмне забезпечення.

2.3. Вибір СУБД

Систему управління базами даних (СУБД) необхідно обирати, беручи до уваги специфіку розроблюваного веб-застосунку. Для даного дипломного проекту доцільно обирати нереляційну СУБД з таких причин:

- Необхідно зберігати велику кількість неструктурованих даних. Нереляційні бази даних не накладають обмежень на типи даних, а також надають можливість додавати нові типи при необхідності.
- Для зберігання медіа-контенту необхідно використовувати окреме сховище. Нереляційні БД дуже зручно використовувати саме в таких випадках.
- Нереляційні БД краще піддаються масштабуванню, що дуже важливо при розробці систем з великою кількістю збережених даних.

2.3.1. *MongoDB*

MongoDB – одна з найбільш відомих нереляційних баз даних, що володіє багатьма функціональними можливостями [17]. Вона є документо-орієнтованою, дані зберігаються у вигляді JSON-документа.

Система може працювати з набором реплік, тобто містити дві або більше копії даних на різних вузлах, що дуже добре впливає на продуктивність системи. Усі операції читання та запису виконуються основною реплікою, однак допоміжні репліки підтримують копії даних в актуальному стані.

Варто зазначити, що MongoDB підтримує горизонтальне масштабування, використовуючи техніку сегментування об’єктів БД.

Дана база даних має багато переваг. Зокрема, вона основана на колекціях різних документів, в яких кількість полів, їх зміст і розмір може відрізнятися один від одного. Кожний об’єкт має зрозумілу структуру.

На сьогоднішній день MongoDB є одною з найбільш часто використовуваних баз даних, яка дозволяє швидко і якісно працювати з великою кількістю даних та об’єктами різної складності.

2.3.2. *Cassandra*

Cassandra – одне з найперших й найбільш широко використовуваних NoSQL-рішень [18]. Вона має ряд переваг, таких як: масштабованість й надійність, висока пропускна здатність для операцій запису й зчитування, SQL-подібна мова запитів, гнучка схема, а також підтримка реплікації.

Cassandra – база даних типу «ключ-значення», тобто зберігає дані як сукупність відповідних пар, де ключ – унікальний ідентифікатор. І ключі, і значення можуть як простими, так і складними об’єктами.

Маючи високу швидкодію у процесі запису даних, дана БД не показує таких самих гарних результатів у читанні, що безпосередньо є мінусом для розроблюваного веб-застосунку, так як користувач буде робити багато запитів на отримання персональних даних, зокрема медіа-файлів. Також, не можна не зазначити про відсутність підтримки агрегації у Cassandra, що є дуже важливою умовою, адже майже усі сутності у БД пов’язані одна з одною.

Для даного веб-застосунку доцільно обрати базу даних MongoDB, адже порівняно з Cassandra вона має такі переваги як:

- підтримка агрегації;
- можливість змінити сутність в процесі розроблення, легко додати нові поля або видалити старі;
- можливість створення складних моделей, ієрархічно пов’язаних між собою.

2.4. Вибір технологій для розроблення клієнтської частини

У веб-застосунках роль клієнта виконує браузер. Для подання контенту на сторінці використовується HTML, для його стилізації – CSS, а щоб користувач міг інтерактивно взаємодіяти з сервером – JavaScript. Для забезпечення гнучкої клієнтської частини доцільно використати JavaScript-фреймворк. Він не тільки полегшує процес написання коду, створюючи та підтримуючи його структуру, а й містить готові рішення базових задач і проблем на початковому етапі розроблення.

Вимоги до JavaScript-фреймворку:

- наявність якісної документації та широкої спільноти розробників;

- інтеграція з іншими бібліотеками;
- розмежування обробки даних та їх представлення.

Angular

Angular добре підходить для динамічних веб-застосунків, з використанням HTML для статичних веб-сторінок [19]. Даний фреймворк реалізує підхід Model-View-Controller (MVC). В Angular поєднуються декларативні шаблони, впровадження залежностей і двонаправлене зв'язування даних, має компонентну структуру. Однак, на відміну від інших фреймворків, вважається складнішим у освоєнні, так як потребує вивчення багатьох концепцій.

React

React відомий нестандартними рішеннями: реалізацією віртуального DOM, створенням елементів інтерфейсу в JavaScript тощо [20]. Його основною ціллю є надати високу швидкість, простоту й масштабованість. Однак при взаємодії з сервером даний фреймворк потребує складного асинхронного програмування.

Vue.js

Відносно молодий фреймворк, який за останні роки набув популярності серед розробників, він поєднав в собі найкращі риси React і Angular. Vue [21] підтримує інтеграцію сторонніх бібліотек, використовує віртуальний DOM, надає реактивність та компонентну структуру, що дозволяє зменшити об'єм коду та використати один компонент декілька разів.

Усі три вище зазначених фреймворки є найбільш популярними на сьогодні, а отже мають велику підтримку розробників та спільноти. Для розроблення клієнтської частини веб-застосунку було обрано Vue.js як найбільш зрозумілий і доступний за своєю структурою фреймворк, який без проблем дозволяє створити швидкий, гнучкий користувацький інтерфейс з використанням додаткових сторонніх бібліотек.

На сьогоднішній день дуже важливо створювати адаптивну верстку для того, щоб веб-сторінка мала коректне подання на всіх пристроях. Для

продуктивної, швидкої роботи і якісного результату доцільно використати CSS-фреймворк, а також CSS-препроцесор. Він попереджає помилки сумісності браузерів, допомагає писати простіший і менший код, що призводить до кращої продуктивності веб-застосунку.

У якості CSS-фреймворку було обрано Bootstrap 4 [22] як найбільш широко використовуваний. Він постійно оновлюється розробниками, має масштабну бібліотеку готових компонентів (кнопки, модальні форми, підказки тощо) та надає можливість швидко створити адаптивний дизайн, який буде гарно виглядати на екранах усіх поширених розмірів.

Використання CSS-препроцесорів надає багато переваг:

- більш чистий код з можливістю повторного використання та окремими змінними;
- більша швидкість написання коду;
- спрощення підтримки коду;
- організованість, структурованість файлів.

Найкращими препроцесорами вважаються SASS [23] та LESS [24]. Вони дозволяють писати CSS з використанням програмних конструкцій замість статичних правил. Обидва препроцесори забезпечують такі розширення CSS: змінні, вкладені блоки, оператори та функції.

Загалом, SASS є потужнішим інструментом ніж LESS і користується більшою популярністю серед розробників. Він має два синтаксиси:

- sass – в ньому відсутні фігурні дужки, правила відокремлюються переведенням рядка, а вкладені елементи реалізовані за допомогою відступів;
- scss – має CSS-подібний синтаксис.

Як і scss, LESS використовує CSS-подібний синтаксис, що безумовно спрощує процес написання й розуміння коду.

Обидва препроцесори є продуктивними, швидкими, зручними. Питання який з них кращий завжди викликало багато суперечок, однак для даного

проекту було обрано SASS, бо він має більш зрозумілий синтаксис та має більшу підтримку серед розробників.

2.5. Вибір сховища для медіа-файлів

Сховище для медіа-файлів – важлива частина архітектури даного веб-застосунку. Визначення місця зберігання файлів, способу доступу до них є серйозною задачею, адже вона напряму впливає на продуктивність роботи.

Зберігання файлів безпосередньо в базі даних вважається поганою практикою, адже це в першу чергу впливає на загальний розмір бази даних, а отже й на час відгуку на запит. Зберігання великих медіа-файлів, таких як зображення або відео, які можуть досягати декількох гігабайтів, очевидно, не є найкращим варіантом.

У розроблюваному веб-застосунку передбачено завантаження великої кількості медіа-файлів різного розміру. Для таких випадків існує два варіанти сховищ:

- веб-сервер;
- хмарне сховище.

Після завантаження файлу у сховище, в обох випадках у базу даних записується відповідне посилання, яке вказує на місцерозташування збереженого файлу.

Локальний веб-сервер

Даний варіант у якості сховища для медіа-файлів є досить популярним рішенням серед розробників.

Він має такий набір переваг:

- контроль над налаштуваннями;
- швидкість передачі даних;
- безпека збережених даних.

Можливість самостійно налаштувати сервер, залежно від потреб веб-застосунку надає декілька варіантів обслуговування файлів, визначення доступу до них, створення найбільш оптимальної ієрархії для забезпечення

швидкого відгуку. Зберігання файлів на локальному сервері надає високу швидкість доступу до них, що є безпосередньою перевагою, особливо при великій кількості даних.

Так як вміст сервера повністю контролюється розробником, і хостингова компанія (у разі використання хмарних сховищ) не може видалити дані або просто розірвати з'єднання, то це гарантує більшу цілісність збережених файлів, а тому й високий рівень безпеки.

Основним недоліком традиційного файлового веб-серверу є складність підтримки. Його складно збільшувати або зменшувати в залежності від росту чи падіння попиту. Також, він має обмежену кількість пам'яті, що означає, що при збільшенні загального обсягу веб-застосунку необхідно розширювати систему.

Хмарне сховище

Хмарне сховище має такий ряд переваг:

- постійна підтримка та оновлення наданого сервісу;
- прості налаштування;
- гнучкість;
- дані зберігаються віддалено.

В першу чергу, постачальник хмарного сховища виконує всі операції з підтримки та оновлення. Це означає більш стабільну роботу та менше ручних налаштувань. Хмарне сховище дозволяє не турбуватися о придбанні серверного простору, підтримки безпеки, виділенні файлового простору або інших задач, пов'язаних з володінням сервера.

Хмарні сховища надають гнучкість. Вони дозволяють збільшувати і зменшувати сервіси в залежності від потреб у реальному часі.

Недоліком даного рішення є втрата контролю, тобто наявність меншої кількості налаштувань. Також, хмарні сервіси частіше стають жертвами кібератак.

Для розроблення даного проекту у було обрано хмарне сховище для зберігання медіа-файлів, адже воно не має обмежень за обсягом, краще підтримується і загалом надає гарну масштабованість.

На сьогодні існує багато сервісів, які надають можливість віддалено зберігати файли. Найпопулярнішими з них вважаються Amazon Web Services (AWS) [25], Microsoft Azure [26] та Google Cloud Platform [27]. Кожен з вищезазначених провідних постачальників має свої сильні та слабкі сторони, які роблять їх гарним вибором для певного проекту.

Amazon Web Services (AWS)

Найсильніша сторона Amazon – домінування на ринку хмарних сховищ. Він являється лідером більше десяти років. Одна з причин його популярності полягає у величезному об’ємі доступних операцій, адже AWS володіє величезним набором доступних послуг. Зокрема, він пропонує довгий список сервісів зберігання, до яких належить Simple Service Storage (S3) для зберігання об’єктів та Elastic File System (EFS) для зберігання файлів.

Велика слабкість Amazon пов’язана з витратами. Не дивлячись на те, що AWS регулярно робить ціни меншими, багатьом розробникам складно керувати витратами, особливо при великих навантаженнях на сервіс.

Microsoft Azure

Головна причина успіху Azure: багато розробників використовують Windows та інше програмне забезпечення Microsoft. Так як Azure тісно інтегрована з цими іншими додатками, то розробники, які використовують велику кількість програмного забезпечення Microsoft вважають, що є сенс використовувати Azure. Основними службами для зберігання даних є сховище BLOB-об’єктів для збереження неструктурованих даних на основі REST, сховище файлів та дискове сховище.

Однак платформа має досить значні недоліки: проблеми з технічною підтримкою та документацією.

Google Cloud Platform (GCP)

Google Cloud Platform спеціалізується на високопродуктивних обчисленнях, таких як великий об’єм даних, аналітика і машинне навчання. Він також пропонує значне балансування масштабування й навантаження. З іншої сторони, Google має не так багато різних сервісів і функцій, як AWS і Azure. Надає хмарне сховище для зберігання об’єктів.

Для даного проекту доцільно обрати хмарне сховище AWS S3, адже даний сервіс надає найбільше можливостей для роботи з файлами різного типу, а також забезпечує захист переданих даних.

2.6. Вибір бібліотеки для роботи з canvas

Canvas – це елемент HTML5, призначений для створення рисунків, анімації, ігор тощо. Він створює область, в якій за допомогою JavaScript можна малювати різні об’єкти, графіки, додавати відео, текст і зображення, трансформувати та змінювати їх властивості.

Canvas ідеально підходить для реалізації інтерактивної веб-дошки, так як володіє усіма засобами для взаємодії з медіа-контентом. Стандартний прикладний програмний інтерфейс canvas низькорівневий, тому створення інтерактивних речей може викликати багато проблем. Саме тому для роботи з даним елементом доцільно використати додаткову бібліотеку.

Бібліотеки canvas надають прикладний програмний інтерфейс, який допомагає більш наглядно створювати об’єкти на полотні, структуровано організовувати взаємодію між ними, а також не витрачати зайвого часу на написання однакового коду для підтримки реактивного подання контенту.

Бібліотека для роботи з canvas повинна володіти такими функціональними можливостями:

- надання засобів для роботи з зображеннями;
- робота з текстом, зокрема його стилізація й редагування;
- підтримка шарів;
- підтримка відслідковування подій.

Так як клієнтська частина веб-застосунку розроблюється за допомогою фреймворку Vue.js, то додатковою, але не основною, вимогою до бібліотеки є можливість інтеграції з Vue.

2.6.1. *Konva.js*

Konva – це бібліотека, яка дозволяє працювати з canvas в об’єктному стилі з підтримкою подій [28]. Вона дозволяє стилізувати та змінювати кожен об’єкт, доданий на полотно. Надає прикладний програмний інтерфейс, підтримує анімації, фільтри, події, серіалізацію та десеріалізацію. Безсумнівною перевагою даної бібліотеки є наявність готових базових об’єктів (прямокутник, коло, текст, полігон, зображення), які мають велику кількість налаштувань, як зовнішніх, так і функціональних можливостей. Інтерактивність виражається у можливості трансформування об’єктів та їх переміщення мишею у будь-яку точку полотна. Konva підтримує створення декількох шарів, які насправді є окремими canvas-об’єктами.

Konva.js можна використовувати разом з Vue.js. Для цього існує окрема бібліотека Vue Konva. Вона забезпечує декларативні та реактивні прив’язки до базової бібліотеки Konva. Усі компоненти Vue Konva відповідають однойменним компонентам Konva, тільки з префіксом «v-».

2.6.2. *Fabric.js*

Fabric.js [29] включає в себе об’єктну модель, SVG-парсер, інтерактивний шар та багато інших інструментів. Бібліотека надає для роботи сім базових фігур і зображення, зовнішній вигляд яких можна змінювати на власний розсуд. Як і Konva, Fabric дозволяє масштабувати, виділяти, рухати, обертати і групувати додані об’єкти.

Значною перевагою Fabric.js є можливість створити два види canvas: статичний і інтерактивний. Перший тип блокує всі функції, які дозволяють взаємодіяти з полотном та його об’єктами. Відповідно другий тип дозволяє проводити вище описані модифікації. Дана перевага може спростити організацію доступу до редагування інтерактивної веб-дошки розроблюваного веб-застосунку.

Fabric.js не має окремої бібліотеки для роботи з Vue.js, проте її можна підключити напряму. Структурна організація коду може виявитися неочевидною, існує вірогідність проблем з продуктивністю і оптимізацією, однак спільна робота Fabric і Vue можлива.

2.6.3. *Paper.js*

Paper.js [30] дозволяє створювати та працювати з векторною графікою. Усі об’єкти розміщуються на шарах, які в поєднанні створюють кінцевий результат. Бібліотека надає широкий спектр функціональних можливостей для роботи з шарами, зокрема різноманітні функції доступу до дочірніх елементів кожного шару.

Paper надає набір базових фігур та кривих для роботи, які можна об’єднувати у групи. Наявне відслідковування подій миші та клавіатури.

Загалом, Paper.js дозволяє створювати анімації, ігри, забезпечує інтерактивну взаємодію з об’єктами, доданими на canvas.

Бібліотека не інтегрується з Vue.js.

Для розроблення даного дипломного проекту було обрано бібліотеку Konva.js, адже, в першу чергу, вона напряму інтегрується з Vue.js, що забезпечить кращу продуктивність, швидкодію веб-застосунку. Konva відповідає усім вище зазначеним вимогам, вона дозволить грамотно організувати ієрархію об’єктів на полотні, створити модулі для роботи з текстом і зображеннями, забезпечити інтерактивність та надати привабливе оформлення веб-дошці і її компонентам.

3. СТРУКТУРНА ОРГАНІЗАЦІЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Опис структур даних веб-застосунку

Для коректної роботи веб-застосунку, його подальшої підтримки, вдосконалення та додавання нових функціональних можливостей необхідно правильно визначити та впровадити структури даних, якими оперує розроблюваний веб-застосунок.

3.1.1. *Сутності веб-застосунку*

В результаті проектування було виявлено такі сутності та їх атрибути:

- Користувач (User):
  - Username – унікальне ім’я користувача. Використовується для авторизації. Повинен мати унікальне значення.
  - Email – використовується для відновлення паролю.
  - Пароль – зберігається у зашифрованому вигляді.
  - Ім’я – повне ім’я користувача, яке буде показуватися на його сторінці.
  - Опис – поле, де користувач може написати інформацію про себе та яке буде показуватися на його сторінці.
  - Аватар.
  - Дошки – список дошок, створених користувачем.
- Дошка (Board):
  - Назва дошки.
  - Стан – показує публічна дошка чи приватна.
  - Автор – унікальний ідентифікатор користувача.
  - Нотатки – список створених нотатків.
  - Зображення – список завантажених зображень.
  - Аудіо – список завантажених аудіо-файлів.
  - Фон – колір фону дошки.
- Медіа (Media):

- Дошка – унікальний ідентифікатор дошки, якій належить медіа-файл.
- Координати – координати, які визначають положення сутності Медіа на дошці.
- Тип – унікальний ідентифікатор, який позначає тип медіа-файлу (текст, зображення чи аудіо).
- Масштаб – зберігає в собі дані про розмір сутності.
- Поворот – зберігає в собі значення кута повороту сутності.

- Нотаток (Note):
  - Медіа – унікальний ідентифікатор сутності Медіа, якій належить дана сутність.
  - Текст.
  - Колір.
- Зображення (Image):
  - Медіа – унікальний ідентифікатор сутності Медіа, якій належить дана сутність.
  - Посилання – посилання на зображення.
  - Фон – фон для оформлення зображення.
  - Назва.
- Аудіо (Audio):
  - Медіа – унікальний ідентифікатор сутності Медіа, якій належить дана сутність.
  - Посилання – посилання на аудіо-файл.
  - Тип – позначає вигляд аудіо на дошці.

Зв’язки між сутностями наведено в табл. 2.

Таблиця 2

Зв’язки між сутностями веб-застосунку

№	Сутність	Сутність	Тип зв’язку
1	User	Board	Один до багатьох
2	Board	Media	Один до багатьох
3	Media	Note	Один до одного
4	Media	Image	Один до одного
5	Media	Audio	Один до одного

Структура описаних сутностей відповідає структурі схеми бази даних. Безпосередньо вони та їх зв‘язки представлено на рис. 1 у вигляді UML діаграми класів.

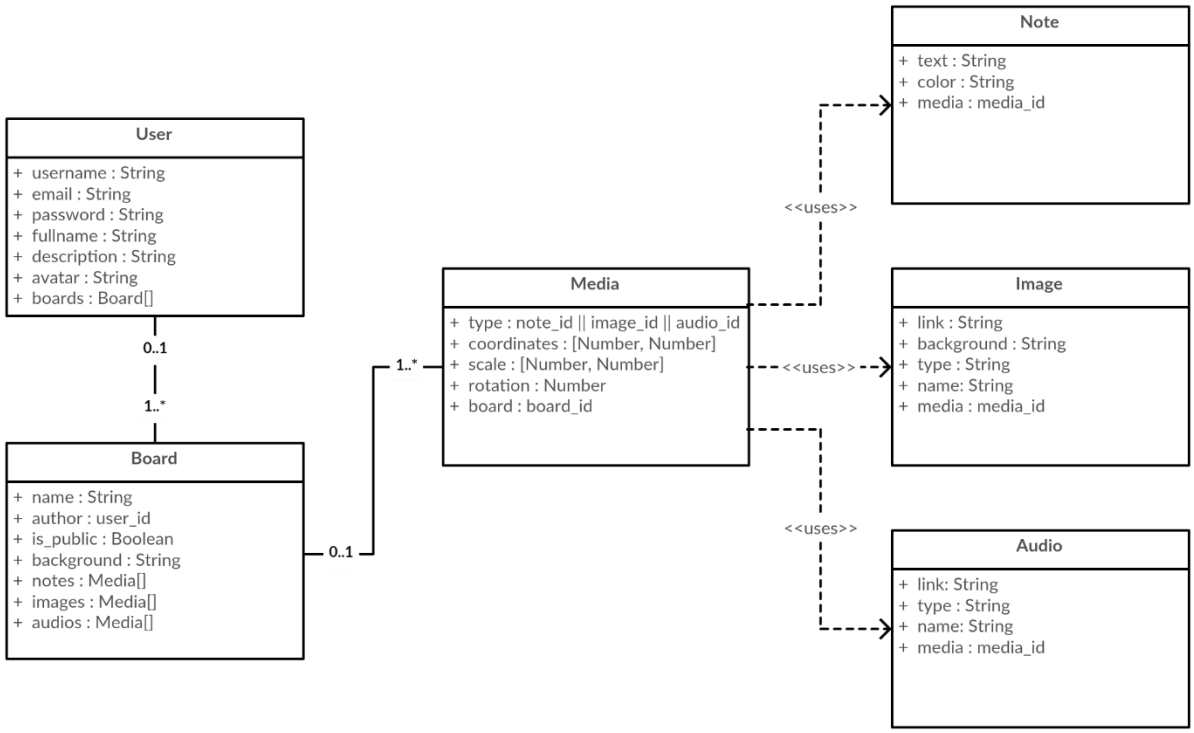


Рис. 1. Структура бази даних веб-застосунку

3.1.2. *Структура сутностей та принципи їх подання на інтерактивній веб-дошці*

Для того, щоб веб-застосунок швидко та правильно показував усі дані на веб-дошці, незалежно від їх фактичної кількості, необхідно правильно організувати початкову структуру кожного об’єкта. Це також важливо задля подальшого розширення веб-застосунку, щоб у майбутньому при додаванні нового атрибута або властивості не потрібно було змінювати базову структуру сутності, що розміщується на дошці.

На дошку можна розмістити три типи об’єктів: текстовий нотаток, зображення та аудіо.

Для реалізації веб-дошки використовується елемент HTML5 canvas, а для безпосередньої роботи з ним обрано бібліотеку Konva.js.

Оскільки Konva дозволяє працювати з canvas в об’єктному стилі, то вона надає набір готових для використання базових об’єктів, зокрема Лінію, Прямокутник, Коло, Зображення, Текст. Кожна сутність на дошці це певний набір базових об’єктів, об’єднаних в одну групу.

Нотаток складається з Тексту та Прямокутника. Прямокутник слугує тілом нотатка, поверх якого накладається Текст, введений користувачем у спеціальному полі для вводу.

Текст має такий набір атрибутів:

- Назва – допомагає у подальшому знайти даний об’єкт на полотні.
- Текст – безпосередньо сам текст нотатку, введений користувачем.
- Розмір шрифту.
- Стиль шрифту.
- Сімейство шрифту.
- Колір тексту.
- Вирівнювання.
- Довжина.
- Ширина.

Нотаток має такий набір атрибутів:

- Назва – допомагає у подальшому знайти даний об’єкт на полотні.
- Колір (обирається користувачем).
- Довжина – базується на довжині тексту.
- Ширина – базується на ширині тексту.
- Колір тіні.
- Прозорість тіні.

Користувач має змогу обрати одне з трьох доступних типів оформлення завантаженого зображення: просте, полароїд або рамка. Кожне з них відрізняється за своєю структурою, однак у будь-якому випадку спочатку необхідно створити новий примірник HTMLImageElement [31], вказати джерело, з якого надходить зображення, а для даного веб-застосунку – це посилання на зображення, що зберігається у хмарному сховищі.

Просте зображення складається з однойменного об’єкта, який має набір базових атрибутів:

- Назва.
- Зображення – примірник HTMLImageElement.
- Довжина.
- Ширина.

Полароїд має більш комплексну структуру з Зображення, Тексту і Прямокутника. Зображення в даному оформленні, окрім вищезазначених, має додаткові атрибути:

- Координата *X*.
- Координата *Y*.
- Колір граней зображення.
- Товщина граней.

Прямокутник зображується тільки білого кольору, його довжина та ширина базується на розмірах Зображення.

Він має такі атрибути:

- Назва.
- Колір граней.
- Товщина граней.
- Колір.
- Довжина.
- Ширина.

Текст має такі самі властивості, як і Текст у Нотатку, проте значення його координати *Y* базується на значенні довжини Зображення.

Тип оформлення у вигляді рамки складається з Зображення та Прямокутника. Вони мають такі самі атрибути, як однойменні об’єкти у типі полароїд. Проте при завантаженні зображення та виборі оформлення користувач має змогу обрати колір рамки з наданого веб-застосунком списку.

Аудіо містить в собі об’єкти Прямокутника, Зображення та двох Текстів. Прямокутник використовується для показу стану аудіо у режимі програвання і має такі атрибути:

- Назва.
- Довжина.
- Ширина.
- Наявність граней.
- Колір граней.
- Товщина граней.

Зображення створюється за таким самим принципом, як описано вище. В даному випадку воно необхідне для візуального подання аудіо-об’єкта на дошці. Користувач може обрати один з трьох типів його оформлення, які відрізняються лише картинкою.

Перший Текст відповідає за назву аудіо, яку користувач записує на власний розсуд при створенні та завантаженні нового об’єкта на дошку. Він має такі самі атрибути, як і його однойменні об’єкти, описані вище. Другий Текст зберігає у собі посилання безпосередньо на аудіо-файл, який

зберігається у хмарному сховищі. Це прихований об’єкт. Він має такі атрибути:

- Назва – допомагає у подальшому знайти даний об’єкт на полотні.
- Текст – зберігає посилання на файл.
- Видимість – завжди має негативне значення.

Вище описано з яких базових об’єктів складається кожна сутність. Усі вони об’єднуються в групи, щоб користувач міг керувати кожною з них як окремим незалежним єдиним цілим. Усі три групи (нотаток, зображення, аудіо) мають однакові властивості та атрибути, проте різні імена, щоб можна було швидко і просто знайти об’єкт певного типу:

- Координата *X*.
- Координата *Y*.
- Назва.
- ID – унікальний ідентифікатор медіа-файлу.
- Кут повороту.
- Масштабування по осі *X*.
- Масштабування по осі *Y*.
- Можливість перетягування.

Також, до атрибутів групи входить функція, яка визначає границі, у межах яких можна перетягувати об’єкт на полотні. Для даного веб-застосунку це межі дошки.

3.2. Узагальнена структура веб-застосунку

Веб-застосунок складається з двох основних частин: серверної та клієнтської. Серверна частина також містить модуль для взаємодії з базою даних та хмарним сховищем.

3.2.1. *Серверна частина*

Серверна частина отримує запити від клієнта, обробляє їх та повертає дані. Вона містить такі засоби:

- обробки HTTP-запитів;

- взаємодії з базою даних;
- взаємодії з хмарним сховищем;
- взаємодії з дошкою (створення, модифікація, видалення);
- авторизації користувачів;
- обробки даних, що надходять від користувача.

Створена за допомогою фреймворку Express на платформі Node.js серверна частина веб-застосунку, окрім базової структури, містить набір додатково встановлених модулів. Базова структура складається з тек src, routes, models і config, її зображено на рис. 2.

У src реалізовано веб-сервер, проведено його базові налаштування і підключено основні модулі для роботи, такі як body-parser [32], morgan [33], cors [34].

Тека routes відповідає за маршрутизацію. Відповідно у файлі users.js оброблюються запити, пов'язані зі сторінками користувача, їх візуальним поданням та налаштуваннями. Також, там реалізовано рішення для реєстрації та авторизації. Для безпечної авторизації користувачів використовуються JSON web token [35] та bcrypt [36]. Перший модуль створює токен доступу, який підписується секретним ключем, та передає його клієнту, який у подальшому використовує цей токен для підтвердження своєї особи. Bcrypt – це адаптивна криптографічна хеш-функція формування ключа, яка використовується для захищеного зберігання пароля.

У файлі boards.js оброблюються запити, які надходять до сервера при роботі з дошкою такі, як додавання нового об'єкта на дошку або його видалення, збереження змін тощо. Файл index.js оброблює запити з головної сторінки веб-застосунку.

На серверній частині веб-застосунку відбувається взаємодія з базою даних MongoDB за допомогою Mongoose – бібліотеки об'єктного моделювання даних [37]. Вона керує зв'язками між даними, забезпечує перевірку схем і використовується для представлення об'єктів у MongoDB. Завдяки Mongoose в окремих JavaScript-файлах було створено схеми

сутностей, зазначених у пункті 3.1.1, визначено зв’язки між ними та додано до них доступ файлам, які відповідають за обробку запитів від користувача.

Медіа-файли зберігаються у хмарному сховищі у окремо виділеному bucket-і, тобто кошику, доступ до якого обмежений. Тека config містить в собі конфігураційний файл з приватними ключами доступу до кошику хмарного сховища, а також іншими приватними налаштуваннями.

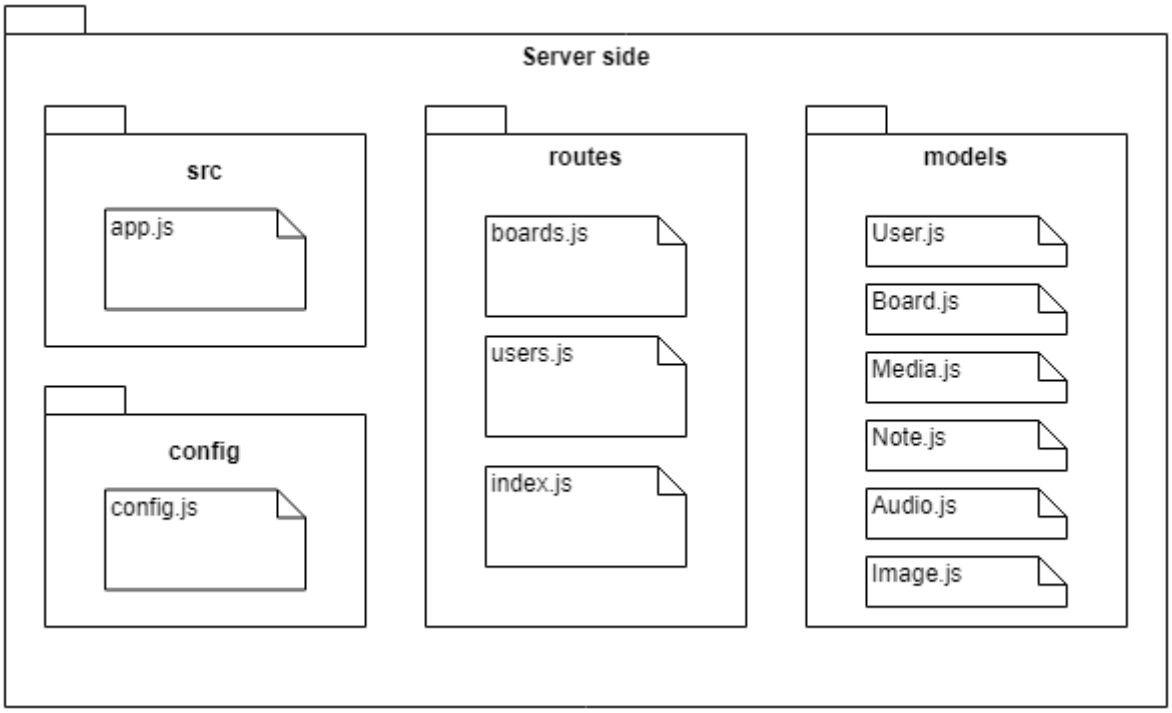


Рис. 2. Структура серверної частини веб-застосунку

3.2.2. *Клієнтська частина*

Клієнтська частина забезпечує взаємодію користувача з веб-застосунком. Вона забезпечує:

- графічний інтерфейс веб-застосунку;
- інтерактивну взаємодію з веб-застосунком без перезавантаження сторінки;
- реактивне оновлення даних на сторінці.

Клієнтська частина складається з таких пакетів: assets, components, router, services, store, styles. Її структуру зображено на рис. 3.

У assets зберігаються статичні файли, такі як зображення та іконки для оформлення інтерфейсу користувача.

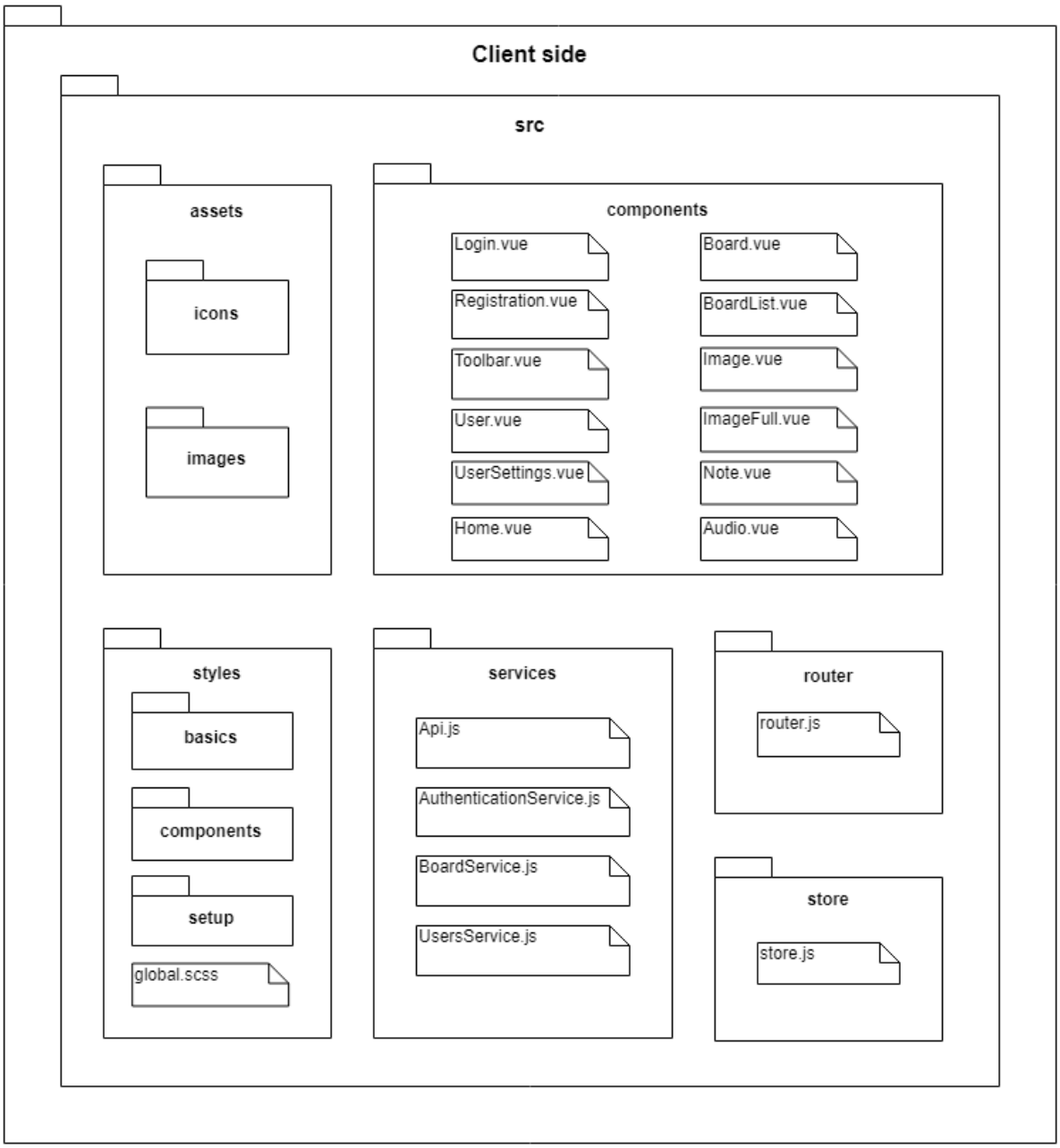


Рис. 3. Структура клієнтської частини веб-застосунку

Фреймворк Vue.js забезпечив гнучку й компонентну структуру веб-застосунку, за що відповідає пакет components.

Кожен компонент є незалежним один від одного і складається з трьох частин:

- HTML-шаблону, який відповідає за подання контенту.

- JavaScript-скрипта, який описує логіку роботи компонента.
- Модульного CSS – стилей, які застосовуються тільки до цього компонента.

Веб-сторінка може поєднувати в собі декілька Vue-компонентів або просто складатися з одного. Така структура допомагає повторно використати готові елементи декілька разів і організовувати гнучку логіку веб-застосунку.

Board – найбільший компонент розроблюваного програмного забезпечення, адже саме він реалізує веб-дошку. Він містить в собі декілька дочірніх компонентів, які розширюють його функціональні можливості, зокрема NoteModal, ImageModal, AudioModal відповідають за завантаження медіа-файлів на дошку, а ImageFull – відкриття та показ зображення у повному розмірі в окремому модальному вікні.

Компоненти Login і Registration реалізовують авторизацію та реєстрацію. Кожен з них містить спеціальну форму, куди користувач вводить свої дані, які надсилаються на сервер за допомогою POST-запиту. Відповідно клієнтська частина отримує відповідь про існування користувача у базі даних веб-застосунку або реєструє його. Користувач не може відправити на сервер некоректні дані, бо кожне поле форми має спеціальну перевірку на правильність введених даних.

Toolbar – навігаційна панель, яка з’являється тільки при успішній авторизації. Вона містить посилання на головну сторінку веб-застосунку, профіль користувача, сторінку зі списком чужих дошок з публічним доступом та кнопку виходу з профіля.

Не надто комплексні компоненти, що відповідають за подання всієї сторінки знаходяться у теці pages. До них відносяться MainPage, UserPage, BoardsPage.

Пакет router надає доступ до параметрів маршруту та дозволяє контролювати навігацію між компонентами.

Пакет services відповідає за взаємодію між клієнтською та серверною частинами. Він використовує JavaScript-бібліотеку Axios для виконання асинхронних HTTP-запитів на сервер.

Використання Axios надає такий набір переваг:

- автоматичне перетворення отриманих даних у формат JSON;
- захист від XSRF-атак на стороні клієнта;
- підтримка Promise API [38].

Authentication Service надає набір функцій, які надсилають на сервер POST запити на реєстрацію та авторизацію користувача, а Board Service надсилає POST та GET запити при взаємодії користувача з дошкою.

Стан веб-застосунку може змінюватись залежно від того чи авторизувався користувач. Для зручного керування станами було налаштовано Vuex [39] для Vue.js. Він слугує централізованим сховищем даних для всіх компонентів застосунку з правилами, які гарантують, що стан може бути змінено тільки передбачуваним чином. Сховище Vuex реактивне. Коли компоненти покладаються на його стан, вони будуть реактивно й ефективно оновлюватися, якщо стан сховища змінюється. Логіку сховища реалізовано у пакеті store.

Для стилізації веб-застосунку було вирішено використовувати CSS-препроцесор SASS. Пакет styles має достатньо комплексну внутрішню структуру, він складається з тек setup, basics, components та файлу global.scss, який містить у собі посилання на усі файли з трьох вище зазначених тек. Головним файлом, що описує стилі є global.scss, саме він компілюється у файл с розширенням .css, який браузер виявляє та використовує для показу веб-сторінки.

Тека setup зберігає файли _normalize.scss та _variables.scss. Перший виправляє помилки і загальні невідповідності браузера, обнуляє усі стандартні налаштування стилів (відступи, розмір шрифту, оформлення тощо). Другий описує змінні, які допомагають зберігати інформацію, яка буде

використовуватися протягом написання усіх стилів проекту. Зокрема, до неї відносяться кольори, шрифти, розмір, стиль тексту тощо.

Тека basics містить файл з базовими стилями для HTML-елементів, які є універсальними для кожної сторінки, коли components описує стилі, унікальні для кожного компонента або сторінки веб-застосунку. Назви файлів у components відповідають назвам Vue-компонентів.

3.2.3. *База даних*

В якості нереляційної бази даних використовується MongoDB. Вона зберігає дані користувача, дошок, які він створює, та інформацію про завантажені медіа-файли. Дана частина веб-застосунку відповідає за такі задачі:

- пошук даних за необхідними параметрами;
- фільтрація даних за необхідними критеріями;
- безпека даних, що зберігаються у базі.

Сутності бази даних, їх опис та зв’язки між ними наведено у пункті 3.1.1.

3.3. Структура веб-дошки та особливості її роботи

Веб-дошка реалізована за допомогою бібліотеки Konva.js.

Головним батьківським компонентом є Сцена (Stage), яка обов’язково повинна містити в собі один або декілька шарів – Layer. В свою чергу кожен шар має два canvas-рендера: рендера сцени та графічного рендера. Рендер сцени – це те, що бачить користувач, а графічний рендер – спеціально прихований canvas, який використовується для високопродуктивного виявлення подій.

Кожен шар може містити об’єкти, групи об’єктів або групи інших груп. Сцена, шари, групи та об’єкти – це віртуальні вузли, схожі на DOM-вузли на HTML-сторінці.

Усі вузли можуть бути трансформовані та візуально змінені. Konva не тільки надає базовий набір об’єктів таких як прямокутник, лінія, коло, текст, зображення тощо, вона дозволяє створити об’єкт власної форми.

Після того як було створено сцену, додано хоча б один шар та об’єкт, можна прив’язувати слухачів подій (event listeners), трансформувати вузли, запускати анімацію та інше.

У розроблюваному веб-застосунку сцена адаптивно змінює свій розмір, залежно від розміру екрану. Вона містить один шар, на якому розміщені усі об’єкти, додані на дошку. Детальну ієрархію вузлів зображено на рис. 4.

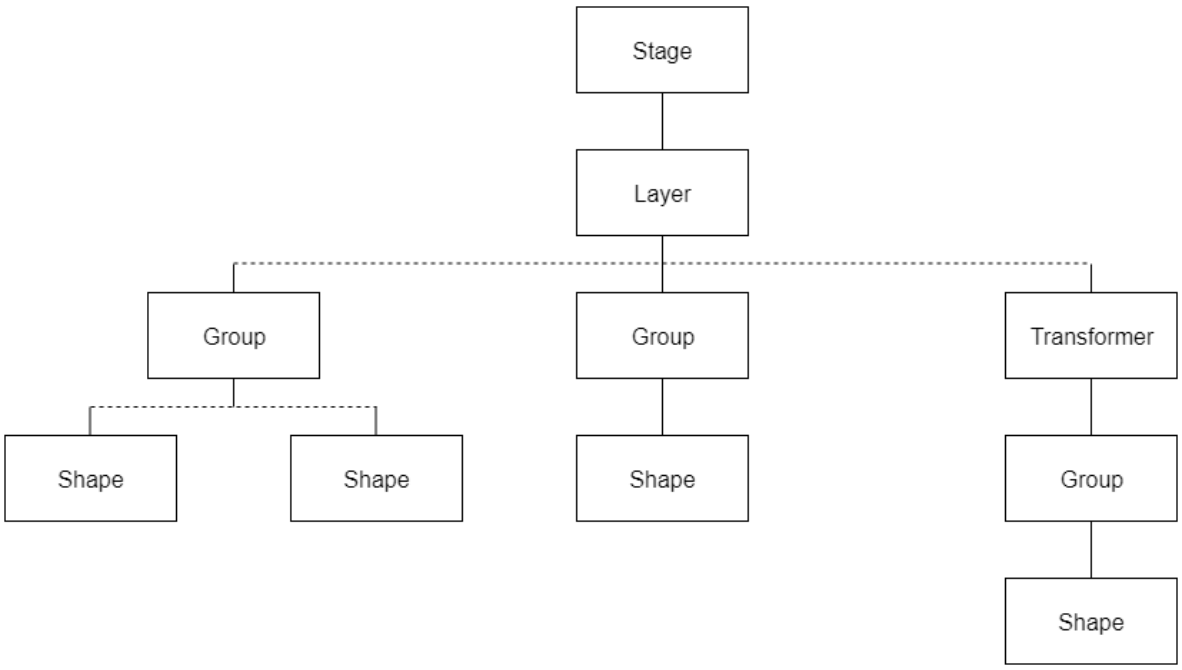


Рис. 4. Ієрархія вузлів сцени

Інтерактивні можливості веб-дошки реалізовано за допомогою властивості Drag-and-Drop та групи Transformer. За допомогою миші користувач може перетягувати об’єкт у будь-яке місце у межах дошки. Konva має вбудовану підтримку перетягування, необхідно лише задати об’єкту властивість draggable.

Transformer – це особливий тип групи, який дозволяє трансформувати об’єкти. Він не змінює довжину та ширину обраного вузла, він змінює його масштаб, дозволяючи зробити його більшим або меншим. Для даного проекту базовий вигляд та властивості Трансформеру було змінено. До цього вузла було додано іконку налаштувань, яка при натисканні динамічно відкриває або

закриває список дій, які можна виконати з об’єктом, а саме: видалити, відкрити зображення у повному розмірі, перемістити об’єкт вгору або вниз.

У розроблюваному веб-застосунку відбувається відстежування події натискання лівої миші на сцену, щоб динамічно додавати Трансформер до об’єкта на дошці. Тоді можна змінювати його розмір та крутити. Щоб прибрати Трансформер необхідно просто натиснути на пусте місце на дошці.

Konva не має вбудованих можливостей для роботи з аудіо-файлами, їх завантаження, прослуховування тощо. Для візуального подання завантаженого медіа-файлу на дошці використано стандартні засоби Konva, описані у пункті 3.1.2.

Програвання та зупинка аудіо реалізовані за допомогою стандартних засобів JavaScript. Зокрема, для створення аудіо-об’єкта викликається відповідний конструктор з параметром URLString, що слугує джерелом, з якого надходить аудіо-файл. У даному випадку це посилання на файл у хмарному сховищі. При подвійному натисканні лівою клавішою миші на аудіо-об’єкті почнеться програвання музики, про що буде сигналізувати відповідна рамка навколо нього. Зупинити програвання можна також подвійним натисканням.

4. АНАЛІЗ РОЗРОБЛЕНОГО ВЕБ-ЗАСТОСУНКУ

4.1. Дизайн та макет сторінок

Дизайн веб-застосунку виконаний у світлих кольорах. Фон сторінок переважно білого кольору, однак усюди використовуються три головні акцентні кольори світлих ніжних відтінків: зелений, помаранчевий і рожевий. Вони підкреслюють творче спрямування веб-застосунку та допомагають акцентувати увагу на важливих деталях. Наприклад, кнопки для підтвердження певних дій мають зелений колір, а для скасування – рожевий.

Макет головної сторінки складається з чотирьох частин. Перша частина містить посилання на сторінку реєстрації та авторизації та великий заголовок з назвою веб-застосунку. Вона займає весь простір екрану, акцентуючи увагу на назві та фоновому зображенні, яке підкреслює призначення програмної розробки – зберігання особистих творчих надбань. Друга частина описує веб-застосунок, головну мету розроблення та розповідає про його загальне призначення. Далі детальніше показано, які основні функції доступні користувачу при взаємодії з застосунком, а саме завантаження трьох типів медіа-файлів, інтерактивна взаємодія з ними тощо. В останньому розділі макету знаходиться зображення з прикладом самої веб-дошки.

Сторінки реєстрації та авторизації мають однаковий принцип. В середині кожної з них знаходиться форма, куди необхідно ввести персональні дані відповідно для створення профілю або доступу до персональної сторінки.

Після успішної авторизації або реєстрації, тобто коли користувач увійшов у систему, на усіх сторінках зверху з’являється навігаційна панель. Вона містить посилання на головну сторінку веб-застосунку, персональну сторінку авторизованого користувача, сторінку перегляду чужих публічних дошок. У правому верхньому кутку знаходиться кнопка виходу з профілю.

Персональна сторінка користувача поділена на дві головні частини: зверху – інформація про користувача, знизу – список його дошок. Після

реєстрації користувач не має жодної дошки, тому у другій половині знаходиться тільки кнопка для створення нової дошки.

Сторінка одночасного редагування та перегляду дошки найбільш комплексна. Як і весь веб-застосунок, вона виконана у світлих кольорах, переважно білому та трьох акцентних: зеленому, рожевому, помаранчевому.

Сама веб-дошка розміщена в центрі сторінки майже на всю висоту екрану. Зверху в центрі знаходиться поле з назвою дошки, яке можна змінити. Поруч справа розміщений набір кольорових кнопок, які змінюють фоновий колір дошки.

Зліва від дошки знаходиться панель з трьома великими кнопками для додавання медіа-файлів. Справа панель для фільтрації контенту, кнопка для зміни статусу дошки з приватної на публічну і навпаки, кнопка для збереження стану та вигляду дошки. Приклад сторінки з веб-дошкою зображено на рис. 5.

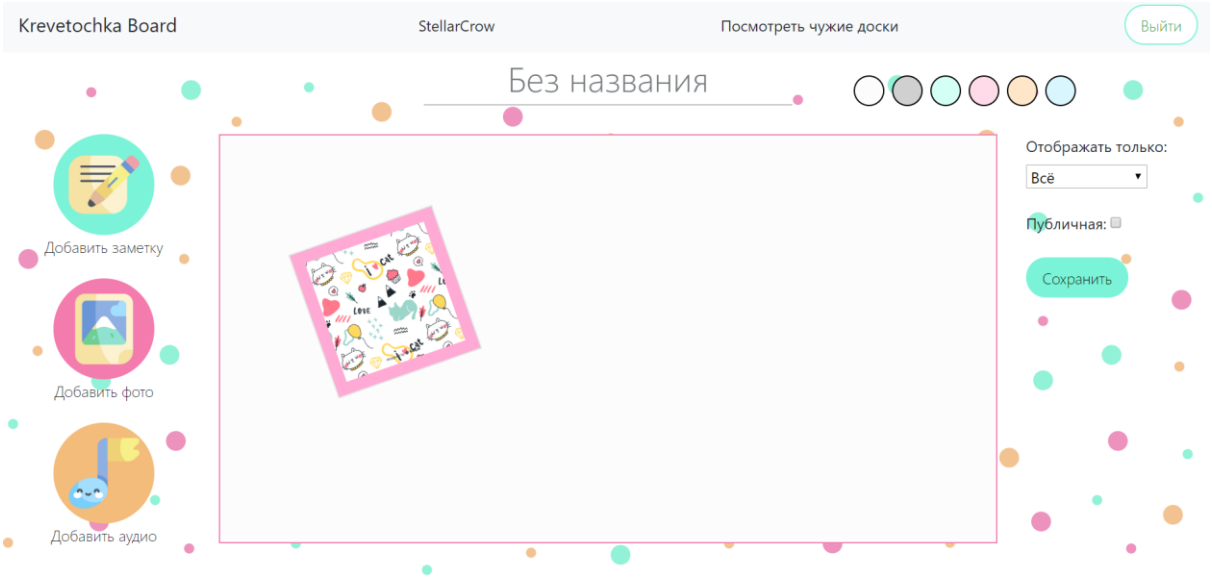


Рис. 5. Приклад сторінки редагування та перегляду веб-дошки

4.2. Тестування веб-застосунку

Тестування є невід'ємною частиною процесу розроблення будь-якого програмного забезпечення. Воно допомагає об'єктивно оцінити якість, коректність роботи, перевірити чи придатний програмний продукт до використання.

Одним зі способів тестування було обрано ручне тестування для перевірки введених даних та створення стресових ситуацій. Воно полягало у введенні неправильних даних в текстові поля, підтвердженні пустих форм реєстрації та авторизації. Працездатність дошки було перевірено такими способами:

- додавання медіа-файлів великих розмірів;
- додавання декількох сотень текстових нотатків;
- додавання декількох сотень зображень різного розміру;
- додавання декількох сотень аудіо різного розміру;
- додавання декількох сотень різних медіа-файлів.

За результатами цього тестування на клієнтську частину веб-застосунку було додано повідомлення про можливі помилки введення некоректних даних у текстові поля форм та обмежено кількість елементів, які можна розмістити на дошці. Це було зроблено не через пониження продуктивності веб-застосунку (веб-дошка працює стабільно при великих навантаженнях), а з точки зору зручності. Переважна більшість екранів візуально не дозволяє розмістити на дошці декілька сотень медіа-файлів. Вони накладаються один на одного, і виникає плутанина. Саме тому веб-застосунок дозволяє створити декілька дошок, якщо користувачу не вистачає місця на одній.

Після вище описаного тестування було встановлено обмеження на розмір медіа-файлів до 5 ГБ. Зважаючи на те, що аудіо та зображення в середньому не великих розмірів, до 1 ГБ, дане обмеження є логічним. Під час завантаження великих медіа-файлів, які можуть бути показані на дошці не одразу, на клієнтській стороні виводиться спеціальне попередження.

Для тестування функціональних можливостей веб-застосунку було розглянуто декілька найбільш використовуваних методів, одним з яких обрано димове тестування [40]. На основі вимог, визначених у пункті 1.1.3 сформовано тестові випадки, зокрема сценарії димового тестування. Було протестовано роботу окремих модулів веб-застосунку та взаємодію між ними.

Для тестування використано операційну систему Windows 10, браузер Google Chrome версії 74.0.3729.169. Точка входу для першого тест-кейсу: запущено браузер, відкрита порожня вкладка. Усі тест-кейси перевіряються послідовно, тому початковими даними для тест-кейсу з номером N є результат, який було отримано після виконання тест-кейсу з номером N–1, крім N=1.

Сценарій димового тестування:

1. Перевірити наявність веб-застосунку за відповідною адресою.
2. Перевірити можливість реєстрації.
3. Перевірити можливість виходу з системи.
4. Перевірити можливість авторизації.
5. Перевірити можливість створення нової дошки.
6. Перевірити додавання нового нотатку на дошку.
7. Перевірити завантаження нового зображення та його додавання на дошку.
8. Перевірити завантаження нового аудіо та його додавання на дошку.
9. Перевірити можливість прослуховування аудіо.
10. Перевірити можливість зупинити програвання аудіо.
11. Перевірити видалення медіа-файлу з дошки.
12. Перевірити можливість відкрити зображення у повному розмірі.
13. Перевірити можливість перенесення об’єкта на передній план.
14. Перевірити можливість перенесення об’єкта на задній план.
15. Перевірити можливість завантажити медіа-файл на комп’ютер.
16. Перевірити можливість збереження стану дошки.
17. Перевірити перегляд дошки іншого користувача.

4.3. Порівняння розробки з існуючими аналогами

У підрозділі 1.2 було проаналізовано існуючі аналоги розробленого веб-застосунку. Майже в усіх існуючих рішеннях відсутня інтерактивність та передбачено роботу лише з одним типом медіа-файлів, а деякі з них мають лише професійне спрямування. Лише Jamboard має схожий набір

функціональних можливостей, тому доцільно більш детально порівняти розроблений веб-застосунок з ним.

Дизайн

Jamboard, як і більшість додатків Google, має простий дизайн, виконаний переважно в сірих тонах, але з одним акцентним кольором – помаранчевим.

Розроблений веб-застосунок також має простий дизайн, однак більш яскравий, зокрема завдяки наявності більшої кількості акцентних кольорів. Вони не тільки привертають увагу, але й закликають до дій. Такі принципи оформлення можуть спонукати користуватися застосунком частіше.

Типи об’єктів на дошці

З медіа-файлів Jamboard дозволяє додати текстовий нотаток та зображення. Завантажити зображення можна лише з Google Drive, однак не з локальної файлової системи. Тому, якщо у запропонованому списку зображень немає файлу, який хочеться розмістити на дошці, необхідно спочатку додати його на Google Drive. Також, на Jamboard є інструмент перо, яке дозволяє малювати на дошці.

Розроблений веб-застосунок працює з більшою кількістю типів медіа-файлів, а саме з текстовими нотатками, зображеннями та аудіо. Останні два можна завантажити з локальної файлової системи. Проте на дошці не можна малювати.

Фільтрація контенту

Коли на дошці знаходиться велика кількість об’єктів різного типу може виникнути плутанина, особливо якщо необхідно знайти якийсь конкретний об’єкт. При великій завантаженості дошки, фільтрація може стати у нагоді.

У Jamboard неможна фільтрувати контент, тобто на полотні завжди показуються всі об’єкти і нічого не можна приховати.

Розроблюваний веб-застосунок дозволяє обирати які елементи показувати на дошці: всі, тільки текст, тільки зображення або тільки аудіо.

Зовнішнє оформлення об’єктів

Зображення у Jamboard не мають ніякого оформлення. Вони розміщуються на дошку у малому масштабі. При створенні текстового нотатка можна обрати його колір.

У створеному веб-застосунку також можна обирати колір нотатка, однак на дошці він відображається зі спеціальною кнопкою, яка візуально закріплює віртуальний папірець на полотні. При завантаженні нового зображення є можливість обрати один з трьох варіантів його оформлення: звичайний, полароїд або кольорова рамка. При завантаженні аудіо можливо обрати один з трьох кольорів ноти, у вигляді якої воно буде розміщено на дошці.

Функціональні можливості

І інтерактивна веб-дошка, і Jamboard однаково дозволяють зробити дошку приватною або публічною, змінити колір робочого полотна, та без обмежень взаємодіяти з медіа-контентом. Об’єкти можна переміщати у будь-яку точку, накладати один на одного, трансформувати, крутити. Кожен медіа-файл в обох рішеннях має власне меню, в якому наведено доступні функції для додаткових перетворень.

Окрім завантаження медіа-файлів та додавання їх на полотно, розглянуті програмні засоби мають додаткові функціональні можливості.

Jamboard надає такий список функцій:

1) Видалення об’єкта з дошки.

2) Копіювання об’єкта на дошці.

3) Можливість змінити текст нотатку.

Додаткові функції інтерактивної веб-дошки:

1) Видалення об’єкта з дошки.

2) Можливість відкрити зображення у повному розмірі.

3) Можливість завантажити медіа-файл на комп’ютер.

4) Перенесення об’єкта на передній план.

5) Перенесення об’єкта на задній план.

Отже, згідно з вище описаним порівнянням двох систем, розроблений веб-застосунок краще підходить для зберігання особистого пам’ятного медіа-контенту. Він надає можливість працювати з різними типами медіа-файлів. Завдяки більш яскравому та привабливому дизайну він привертає увагу дітей та молоді, а різноманітні варіанти оформлення медіа-контенту дозволяють проявити, розвинути творчі здібності та зробити кожну дошку по-справжньому унікальною. Можливість фільтрації контенту може допомогти при пошуку об’єкта, якщо на дошці розміщено багато медіа-файлів.

4.4. Рекомендації щодо подальшого вдосконалення

Розроблений веб-застосунок відповідає всім вимогам, які було визначено під час аналізу предметної області. Реалізовано усі функціональні можливості. Однак розроблену версію інтерактивної веб-дошки для зберігання особистого медіа-контенту планується вдосконалювати в майбутньому. Серед подальших змін планується:

- вдосконалити роботу веб-застосунку для браузерів Microsoft Edge, Firefox, Internet Explorer;
- додати більше варіантів оформлення завантажених на дошку медіа-файлів;
- додати можливість змінювати оформлення розміщених на дошці об’єктів;
- вдосконалити можливості комунікації між користувачами на кшталт надання змоги коментувати та оцінювати чужі дошки;
- додати можливість завантажити та запустити відео-файл;
- додати можливість завантажити великий текстовий файл;
- додати можливість завантажити медіа-файл розміром більше, ніж 5 ГБ.

ВИСНОВКИ

Метою даного проекту є створення інтерактивної веб-дошки для зберігання особистого медіа-контенту. В результаті порівняння переваг і недоліків мобільних додатків та веб-застосунків найкращим рішенням прийнято реалізацію проекту у вигляді веб-застосунку. У рамках дипломного проекту визначено вимоги до розроблюваної системи та оглянуто існуючі рішення, в результаті чого виявлено, що існуючі програмні рішення не задовольняють поставленим вимогам, що вказує на необхідність розроблення веб-застосунку для зберігання особистого медіа-контенту з інтерактивними можливостями та творчим спрямуванням.

Обґрунтовано вибір технологій для розроблення веб-застосунку, зокрема найкращими рішеннями є:

- платформа Node.js – для серверної частини;
- фреймворк Vue.js – для клієнтської частини;
- MongoDB – СУБД.

Розроблений веб-застосунок дозволяє створити власний профіль, створити декілька дошок, додати на дошку текстовий нотаток, зображення, аудіо. Усі додані медіа-файли можна крутити, розташовувати в будь-якому місці у межах дошки, змінювати їх розмір, видаляти, переміщати на передній і задній план. Зображення відкриваються у повному розмірі, а аудіо програються після подвійного натискання.

Було успішно проведено тестування розробленого веб-застосунку. Усі функціональні можливості відповідають поставленим вимогам.

Інтерактивну веб-дошку для зберігання медіа-контенту планується вдосконалювати в майбутньому, зокрема додати можливість працювати з відео-файлами, оптимізувати роботу для різних браузерів та додати більше варіантів оформлення медіа-контенту.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Mobile vs Desktop : [Електронний ресурс]. Режим доступу: https://www.stonetemple.com/mobile-vs-desktop-usage-study/
2. Forecast unit shipments of tablets worldwide from 2010 to 2023 (in million units) : [Електронний ресурс]. Режим доступу: https://www.statista.com/statistics/269912/worldwide-tablet-shipments-forecast/
3. Behance : [Електронний ресурс]. Режим доступу: https://www.behance.net/
4. Pinterest : [Електронний ресурс]. Режим доступу: https://www.pinterest.com/
5. Instagram : [Електронний ресурс]. Режим доступу: https://www.instagram.com/
6. Tumblr : [Електронний ресурс]. Режим доступу: https://www.tumblr.com/
7. Jamboard : [Електронний ресурс]. Режим доступу: https://gsuite.google.com/products/jamboard/
8. Google Drive : [Електронний ресурс]. Режим доступу: https://www.google.com/drive/
9. Python : [Електронний ресурс]. Режим доступу: https://www.python.org/
10. Ruby : [Електронний ресурс]. Режим доступу: https://www.ruby-lang.org/ru/
11. JavaScript Documentation : [Електронний ресурс]. Режим доступу: https://devdocs.io/javascript/
12. Node.js : [Електронний ресурс]. Режим доступу: https://nodejs.org/en/
13. Express : [Електронний ресурс]. Режим доступу: https://expressjs.com/
14. Koa : [Електронний ресурс]. Режим доступу: https://koajs.com/
15. Sails.js : [Електронний ресурс]. Режим доступу: https://sailsjs.com/

16. Waterline ORM. Sails documentation : [Електронний ресурс]. Режим доступу: https://sailsjs.com/documentation/reference/waterline-orm
17. MongoDB : [Електронний ресурс]. Режим доступу: https://www.mongodb.com/
18. Cassandra : [Електронний ресурс]. Режим доступу: http://cassandra.apache.org/
19. Angular : [Електронний ресурс]. Режим доступу: https://angular.io/
20. React : [Електронний ресурс]. Режим доступу: https://reactjs.org/
21. Vue.js : [Електронний ресурс]. Режим доступу: https://vuejs.org/
22. Bootstrap 4 : [Електронний ресурс]. Режим доступу: https://getbootstrap.com/
23. SASS : [Електронний ресурс]. Режим доступу: https://sass-lang.com/
24. LESS : [Електронний ресурс]. Режим доступу: http://lesscss.org/
25. Amazon Web Services : [Електронний ресурс]. Режим доступу: https://aws.amazon.com/ru/about-aws/
26. Microsoft Azure : [Електронний ресурс]. Режим доступу: https://azure.microsoft.com/ru-ru/
27. What is Google’s Cloud Platform: [Електронний ресурс]. Режим доступу: https://medium.com/@retomeier/what-is-googles-cloud-platform-d92a9c9e5e89
28. Konva.js Features : [Електронний ресурс]. Режим доступу: https://konvajs.org/
29. Fabric.js : [Електронний ресурс]. Режим доступу: http://fabricjs.com/
30. Paper.js. Features : [Електронний ресурс]. Режим доступу: http://paperjs.org/features/
31. HTMLImageElement. JavaScript Documentation : [Електронний ресурс]. Режим доступу: https://www.javascripture.com/HTMLImageElement
32. Body-parser : [Електронний ресурс]. Режим доступу: https://www.npmjs.com/package/body-parser

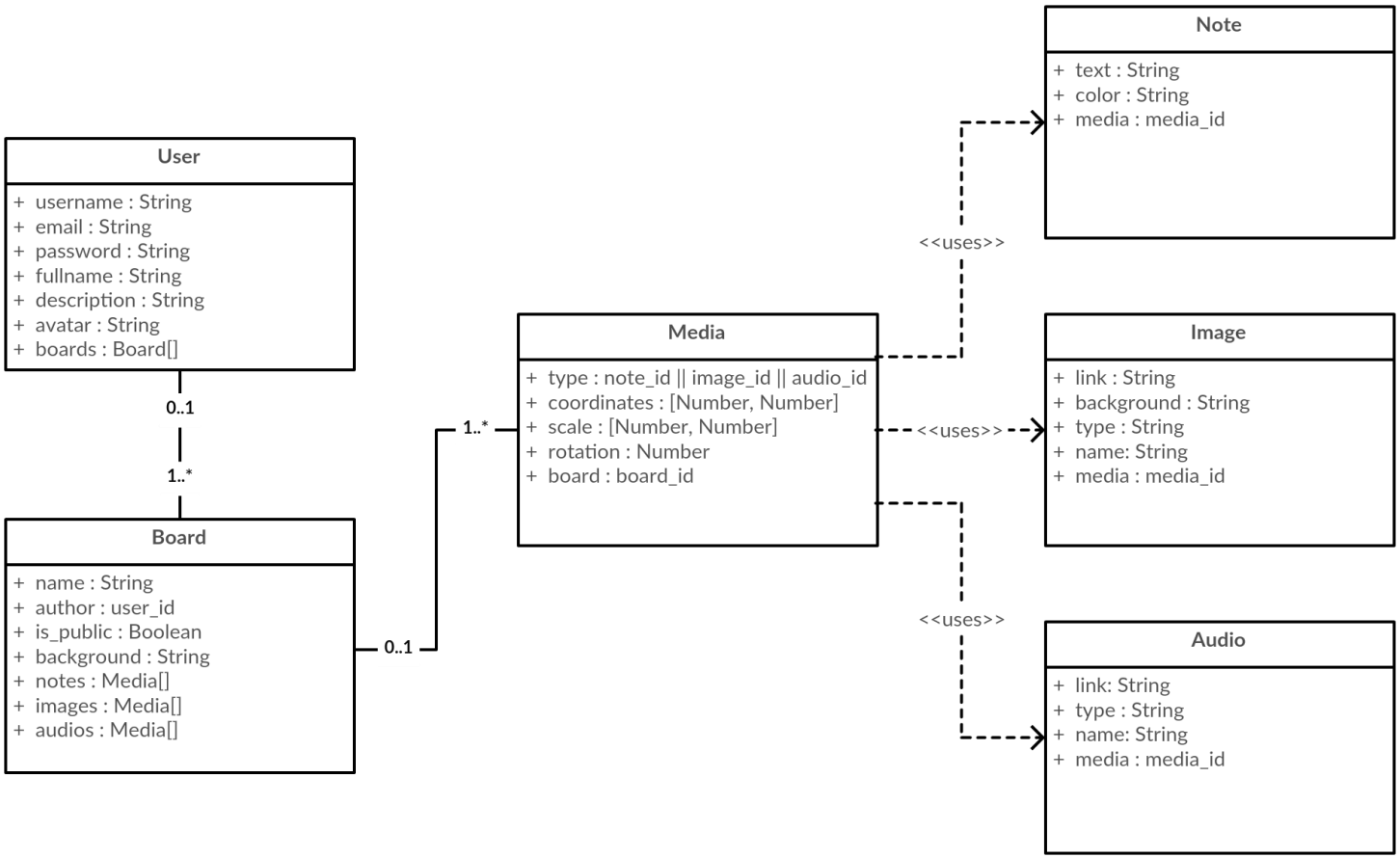
33. Morgan : [Електронний ресурс]. Режим доступу: https://www.npmjs.com/package/morgan
34. Cors : [Електронний ресурс]. Режим доступу: https://www.npmjs.com/package/cors
35. Про токены, JSON Web Tokens (JWT), аутентификацию и авторизацию. Token-Based Authentication: [Електронний ресурс]. Режим доступу: https://gist.github.com/zmts/802dc9c3510d79fd40f9dc38a12bccfc
36. Hashing in Action: Understanding bcrypt : [Електронний ресурс]. Режим доступу: https://auth0.com/blog/hashing-in-action-understanding-bcrypt/
37. Mongoose : [Електронний ресурс]. Режим доступу: https://mongoosejs.com/
38. Promise API: [Електронний ресурс]. Режим доступу: https://javascript.info/promise-api
39. What is Vuex?: [Електронний ресурс]. Режим доступу: https://vuex.vuejs.org/
40. Smoke Testing : [Електронний ресурс]. Режим доступу: http://softwaretestingfundamentals.com/smoke-testing/

ДОДАТКИ

Додаток 1

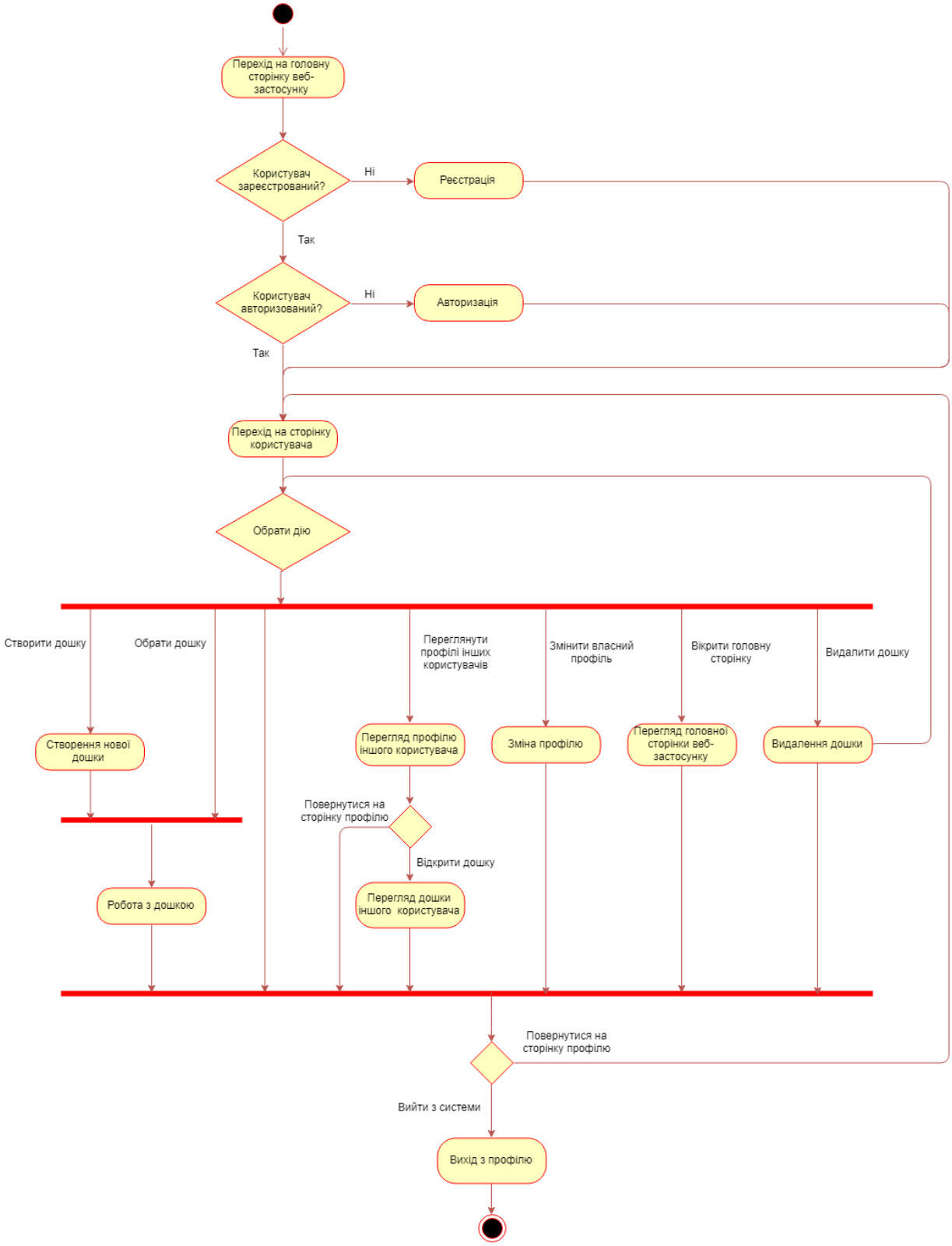
Копії графічних матеріалів

ДП.045440-06-99. Інтерактивна веб-дошка для зберігання особистого медіа-контенту. Функціональні можливості веб-застосунку. Діаграма використання



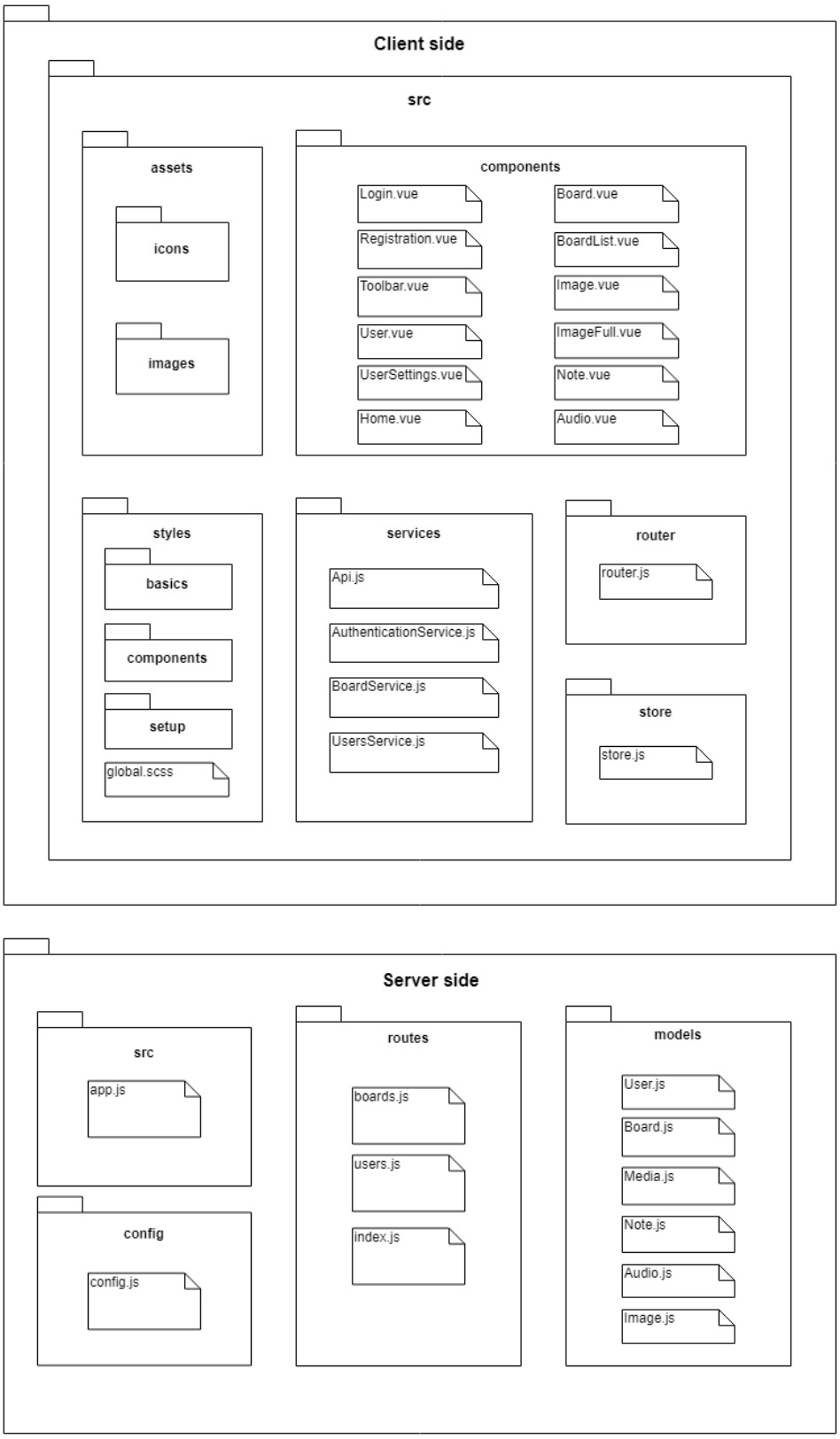
ДП.045440-07-99. Інтерактивна веб-дошка для зберігання особистого медіа-контенту. Архітектура веб-застосунку. Схема бази даних

Діаграма діяльності



Дячук Д.С., група КП-51

Структура серверної та клієнтської частини веб-застосунку



Дячук Д.С., група КП-51

Додаток 2
Копія презентації

ІНТЕРАКТИВНА ВЕБ-ДОШКА ДЛЯ ЗБЕРІГАННЯ ОСОБИСТОГО МЕДІА-КОНТЕНТУ

Виконала: Дячук Д.С., КП-51

Науковий керівник: старший викладач кафедри ПЗКС, Гадиняк Р.А.

Актуальність

Існуючі рішення

Behance

Tumblr

Instagram

Pinterest

Jamboard

Недоліки існуючих рішень

- Відсутність інтерактивності;
- Робота переважно з одним типом медіа-файлів;
- Цінні файли губляться у великому потоці даних;
- Професійне спрямування.

Постановка задачі

- **Мета проекту**: забезпечити зберігання особистих унікальних творчих надбань у формі медіа-файлів шляхом розроблення програмного забезпечення з інтерактивними можливостями.
- **Задачі проекту**:
  - Проаналізувати існуючі рішення;
  - Обрати засоби реалізації програмного забезпечення;
  - Розробити програмне забезпечення;
  - Протестувати розроблене програмне забезпечення.

Вимоги

- Можливість створити декілька дошок.
- Видалити дошку.
- Додати на дошку текст.
- Додати на дошку зображення.
- Додати на дошку аудіо.
- Можливість відфільтрувати контент на дошці за типом
- Розмістити об’єкт на дошці у будь-якому місці.
- Змінити розмір та кут повороту об’єкта на дошці.
- Зберегти стан дошки
- Зробити дошку публічною або приватною.

Обрані засоби реалізації


Архітектура веб-застосунку

- Клієнтська частина
- Серверна частина
- База даних
- Хмарне сховище

Структура бази даних

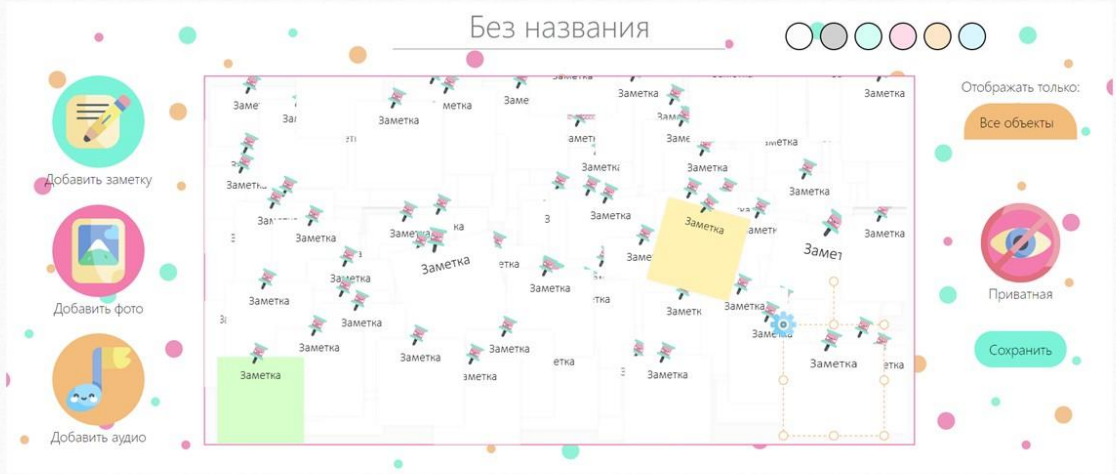
Структура серверної та клієнтської частини

Особливості веб-застосунку

- Перетягування об’єкта на дошці на інший шар
- Завантаження медіа-файла на комп’ютер
- Перегляд зображення з дошки у повному розмірі
- Зміна фонового кольору дошки

Тестування

- Ручне тестування;
- Створення стресових ситуацій;
- Димове тестування.



Порівняння з існуючими рішеннями

	Інтерактивна веб-дошка	Google Jamboard
К-сть типів медіа-файлів	3	2
Оформлення контенту	+	-
Фільтрація контенту	+	-
Відкриття зображення у повному розмірі	+	-
Перенесення об'єкта на інший шар	+	-
Завантаження з дошки на комп'ютер	+	-
Копіювання об'єкта на дошці	-	+

Конференція

- Даний дипломний проект був представлений на четвертій міжнародній науково-технічній конференції «Поліграфічні, мультимедійні і Web-технології» 2019 і опублікований у вигляді тез доповіді.

Висновки

- Проаналізовано існуючі рішення проблеми зберігання медіа-контенту;
- Розроблено веб-застосунок, який в інтерактивній формі допомагає зберегти особисті творчі надбання у вигляді медіа-контенту.
- Протестовано розроблений веб-застосунок;
- Рекомендації щодо вдосконалення:
  - Додати можливість завантажити та запустити відео-файл;
  - Додати можливість завантажити файл більше 5 ГБ;
  - Вдосконалити комунікацію між користувачами;
  - Більше оформлення медіа-файлів.

Дякую за увагу!

Додаток 3
Лістинг програми

```
import NoteModal from './NoteModal'
import ImageModal from './ImageModal'
import AudioModal from './AudioModal'
import ImageFull from './ImageFull'
import BoardService from "../services/BoardService"

const imagePink = require("../assets/icons/music-icon-pink.png");
const imageGreen = require("../assets/icons/music-icon-green.png");
const imageYellow = require("../assets/icons/music-icon-yellow.png");
const notePinIcon = require("../assets/icons/note-pin-icon.png");
const settings = require("../assets/icons/026-setting.png");

let width = window.innerWidth;
let height = window.innerHeight;
export default {
  name: "Board",
  components: {NoteModal, ImageModal, AudioModal, ImageFull},
  data() {
    return {
      bname: 'Без названия',
      colorStage: "#fcfcfc",
      colorsStage: [
        {id: "6", color: "white", num: "#fcfcfc"},
        {id: "7", color: "black", num: "#d0d0d0"},
        {id: "8", color: "green", num: "#d4fff5"},
        {id: "9", color: "pink", num: "#ffdbea"},
        {id: "10", color: "orange", num: "#ffe6c8"},
        {id: "11", color: "blue", num: "#d9f6ff"}
      ],
      author: '',
      is_public: false,
      audioModal: false,
      imageModal: false,
      imageFull: false,
      noteModal: false,
      imageFullFile: '',
      filter: 'all',
      savedMessage: "",
      notes: [],
      images: [],
      audios: [],
      stageLayer: null,
      stageSize: {
        width: width,
        height: height
      },
      transformer: {
        anchorSize: 10,
        anchorCornerRadius: 5,
        anchorStroke: '#F3BC7B',
        borderStroke: '#F3BC7B',
        borderDash: [3, 3]
      },
      selectedShapeId: '',
      selectedAudioId: '',
      audioFile: null
    }
  },
  created: function () {
    window.addEventListener('resize', this.changeRect);
    this.changeRect();
  },
```

```
      methods: {
    async saveBoard() {
            let notesUpdated = [];
            let imagesUpdated = [];
            let audiosUpdated = [];
            const id = this.$store.state.route.params.idb;

            this.notes.forEach(function (note) {
              let updatedNote = {};
              updatedNote.coordinates=
[note.children[0].getAbsolutePosition().x,
note.children[0].getAbsolutePosition().y];
              updatedNote.rotation = note.rotation();
              updatedNote.scale = [note.scaleX(), note.scaleY()];
              updatedNote.id = note.id();
              notesUpdated.push(updatedNote);
            });

            this.images.forEach(function (image) {
              let updatedImage = {};
              updatedImage.coordinates=
[image.children[0].getAbsolutePosition().x,
image.children[0].getAbsolutePosition().y];
              updatedImage.id = image.id();
              updatedImage.rotation = image.rotation();
              updatedImage.scale = [image.scaleX(), image.scaleY()];
              imagesUpdated.push(updatedImage);
            });

            this.audios.forEach(function (audio) {
              let updatedAudio = {
                coordinates:      [audio.children[0].getAbsolutePosition().x,
audio.children[0].getAbsolutePosition().y],
                id: audio.id(),
                rotation: audio.rotation(),
                scale: [audio.scaleX(), audio.scaleY()],
              };
              audiosUpdated.push(updatedAudio);
            });

            let data = {
              idb: id,
              is_public: this.is_public,
              name: this.bname,
              background: this.colorStage,
              notes: notesUpdated,
              images: imagesUpdated,
              audios: audiosUpdated
            };

            const response = await BoardService.saveBoard(data);
            this.savedMessage = "Сохранено " + this.getCurrentDate();
          },
    async noteDataFromModal(data) {
            this.noteModal = data.noteModal;
            if (data.text === "") return;

            const id = this.$store.state.route.params.idb;
            const newNote = await BoardService.createNote({
              boardId: id,
              text: data.text,
              color: data.color,
              coordinates: [20, 20],
              rotation: 0,
```

```
    scale: [1, 1]
  });
  let noteData = {
    id: newNote.data.id,
    text: data.text,
    color: data.color,
    coordinates: [20, 20],
    rotation: 0,
    scale: [1, 1]
  };

  this.createNote(noteData);
},
createNote(data) {
  const stage = this.$refs.stage.getNode();
  let layer = this.stageLayer;
  let group = new Konva.Group({
    x: data.coordinates[0],
    y: data.coordinates[1],
    draggable: true,
    name: 'noteGroup',
    id: data.id,
    rotation: data.rotation,
    scaleX: data.scale[0],
    scaleY: data.scale[1],
    dragBoundFunc: function (pos) {
      let newY, newX;
      if (pos.y < -50) {
        newY = -50;
      } else if (pos.y > stage.height()) {
        newY = stage.height()
      } else
        newY = pos.y;

      if (pos.x < -50) {
        newX = -50;
      } else if (pos.x > stage.width()) {
        newX = stage.width()
      } else
        newX = pos.x;

      return {
        x: newX,
        y: newY,
      }
    }
  });

  let noteText = new Konva.Text({
    name: 'noteText',
    text: data.text,
    fontSize: 14,
    fontStyle: 200,
    fontFamily: 'Calibri',
    fill: '#000',
    width: 100,
    height: 100,
    padding: 15,
    align: 'center'
  });

  if (data.text.length > 25) {
    noteText.fontSize(12);
    noteText.width(140);
```

```
          noteText.height(140);
        }

        let rect = new Konva.Rect({
          name: 'noteRect',
          stroke: "#e5e5e5",
          strokeWidth: 1,
          fill: data.color,
          width: noteText.width(),
          height: noteText.height(),
          shadowColor: '#e5e5e5',
          shadowBlur: 5,
          shadowOpacity: 0.5
        });

        let imageObj = new Image();
        imageObj.src = notePinIcon;

        let notePin = new Konva.Image({
          x: rect.width() / 4,
          y: -10,
          name: 'audioImage',
          image: imageObj,
          width: rect.width() / 3,
          height: rect.height() / 3,
          centeredScaling: true,
          rotation: -20
        });

        rect.cache();
        group.add(rect);
        group.add(noteText);
        group.add(notePin);
        layer.add(group);
        this.notes.push(group);
        stage.batchDraw();
      },
async audioDataFromModal (data) {
        this.audioModal = data.audioModal
        if (data.audioFile === null) return

        const id = this.$store.state.route.params.idb
        let formData = new FormData()
        formData.append('audio', data.audioFile)

        let audio = {
          audioType: data.audioType,
          name: data.name,
          coordinates: [60, 20],
          rotation: 0,
          scale: [1, 1]
        }

        await BoardService.uploadAudio(formData, id)
          .then(response => {
            if (response.data.audioId === -1) return
            return BoardService.createAudio({
              audio: audio,
              bid: id,
              audioId: response.data.audioId
            })
          })
          .then((response) => {
            console.log(response.data.audioLink)
```

```
            audio.link = response.data.audioLink
            audio.id = response.data.media
          })

        this.createAudio(audio)
      },
createAudio (data) {
        let layer = this.stageLayer
        const stage = this.$refs.stage.getNode()

        let group = new Konva.Group({
          x: data.coordinates[0],
          y: data.coordinates[1],
          draggable: true,
          name: 'audioGroup',
          id: data.id,
          rotation: data.rotation,
          scaleX: data.scale[0],
          scaleY: data.scale[1],
          dragBoundFunc: function (pos) {
            let newY, newX;
            if (pos.y < -50) {
              newY = -50
            } else if (pos.y > stage.height()) {
              newY = stage.height()
            } else
              newY = pos.y

            if (pos.x < -50) {
              newX = -50
            } else if (pos.x > stage.width()) {
              newX = stage.width()
            } else
              newX = pos.x

            return {
              x: newX,
              y: newY,
            }
          }
        });

        let imageObj = new Image()
        if (data.audioType === 'icon-green') {
          imageObj.src = imageGreen
        }
        else if (data.audioType === 'icon-pink') {
          imageObj.src = imagePink
        }
        else if (data.audioType === 'icon-yellow') {
          imageObj.src = imageYellow
        }

        let audioRect = new Konva.Rect({
          name: 'audioRect',
          width: 120,
          height: 120,
          strokeEnabled: false,
          stroke: '#f3bc7b',
          strokeWidth: 5
        });

        let audioImage = new Konva.Image({
          name: 'audioImage',
```

```
            image: imageObj,
            width: audioRect.width(),
            height: audioRect.height() - 10,
          });

          let audioName = new Konva.Text({
            y: 110,
            name: 'audioName',
            text: data.name,
            fontSize: 14,
            fontStyle: 200,
            fontFamily: 'Calibri',
            fill: '#000',
            width: audioImage.width()
          });

          let audioText = new Konva.Text({
            name: 'audioText',
            text: data.link,
            visible: false
          });

          group.add(audioRect);
          group.add(audioImage);
          group.add(audioName);
          group.add(audioText);
          layer.add(group);
          this.audios.push(group);
          stage.batchDraw();
        },
async imageDataFromModal(data) {
          this.imageModal = data.imageModal;
          if (data.imageFile == null) return;

          const id = this.$store.state.route.params.idb;
          let image = {};

          image.imageType = data.imageType;
          image.color = data.color;
          image.name = data.name;
          image.coordinates = [40, 20];
          image.rotation = 0;
          image.scale = [1, 1];

          let formData = new FormData();
          formData.append('photo', data.imageFile);

          await BoardService.uploadImage(formData, id)
            .then((response) => {
              if (response.data.imageId === -1) {
                return console.log("imageId -1 " + response.data.imageId);
              }
              console.log("image ID " + response.data.imageId);
              return BoardService.createImage({
                image: image,
                bid: id,
                imageId: response.data.imageId
              })
            })
            .then((response) => {
              console.log(response.data.imageLink);
              image.link = response.data.imageLink;
              image.id = response.data.media;
            });
```

```
        this.createImage(image);
      },
createImage(data) {
        const stage = this.$refs.stage.getNode();
        let layer = this.stageLayer;
        let newImage, rect;
        let group = new Konva.Group({
          x: data.coordinates[0],
          y: data.coordinates[1],
          draggable: true,
          name: 'imageGroup',
          id: data.id,
          rotation: data.rotation,
          scaleX: data.scale[0],
          scaleY: data.scale[1],
          dragBoundFunc: function (pos) {
            let newY, newX;
            if (pos.y < -50) {
              newY = -50;
            } else if (pos.y > stage.height() - 20) {
              newY = stage.height() - 20
            } else
              newY = pos.y;

            if (pos.x < -50) {
              newX = -50;
            } else if (pos.x > stage.width() - 20) {
              newX = stage.width() - 20
            } else
              newX = pos.x;

            return {
              x: newX,
              y: newY,
            }
          }
        });

        let imageObj = new Image();
        imageObj.src = data.link;

        if (data.imageType === "simple") {
          newImage = new Konva.Image({
            name: 'image',
            image: imageObj,
            width: 100,
            height: 100
          });
          group.add(newImage);
        }
        else if (data.imageType === 'polaroid') {

          newImage = new Konva.Image({
            x: 5,
            y: 10,
            name: 'image',
            image: imageObj,
            stroke: "#72787a",
            strokeWidth: 1,
            width: 100,
            height: 100
          });

          let imageText = new Konva.Text({
```

```
y: newImage.height(),
              name: 'imageText',
              text: data.name,
              fontSize: 14,
              fontStyle: 200,
              fontFamily: 'Calibri',
              fill: '#000',
              padding: 20,
              align: 'center'
            });

            rect = new Konva.Rect({
              name: 'imageBack',
              stroke: "#e5e5e5",
              strokeWidth: 1,
              fill: data.color,
              width: newImage.width() + 10,
              height: newImage.height() + imageText.height()
            });

            imageText.width = rect.width;
            group.add(rect);
            group.add(newImage);
            group.add(imageText);
          }
          else if (data.imageType === 'frame') {
            newImage = new Konva.Image({
              x: data.coordinates[0] + 10,
              y: data.coordinates[1] + 10,
              name: 'image',
              image: imageObj,
              width: 100,
              height: 100
            });

            rect = new Konva.Rect({
              x: data.coordinates[0],
              y: data.coordinates[1],
              name: 'imageBack',
              stroke: "#e5e5e5",
              strokeWidth: 1,
              fill: data.color,
              width: newImage.width() + 20,
              height: newImage.height() + 20
            });

            group.add(rect);
            group.add(newImage);
          }

          layer.add(group);
          this.images.push(group);
          stage.batchDraw();
        },
  parseDataFromServer(data) {
          let notes = data.notesArray;
          let images = data.imagesArray;
          let audios = data.audiosArray;
          let board = data.board;
          this.bname = board.bname || 'Без названия';
          this.is_public = board.is_public;
          this.colorStage = board.background || '#fcfcfc';
          this.author = board.author;
```

```
        this.stageLayer = this.$refs.layer.getNode();

        for (let i = 0; i < notes.length; i++) {
          this.createNote(notes[i]);
        }

        for (let j = 0; j < images.length; j++) {
          this.createImage(images[j]);
        }

        for (let i = 0; i < audios.length; i++) {
          this.createAudio(audios[i]);
        }
      },
handleStageMouseDown(e) {
        let selectedGroup;

        if (e.target === e.target.getStage()) {
          this.selectedShapeId = '';
          this.updateTransformer();
          return;
        }

        const clickedOnTransformer =
          e.target.getParent().className === 'Transformer';
        if (clickedOnTransformer) {
          return;
        }

        const targetId = e.target.getParent().id();
        this.stageLayer.children.forEach(function (group) {
          if (group.id() === targetId) {
            selectedGroup = group;
          }
        });

        if (selectedGroup) {
          this.selectedShapeId = targetId;
        }
        else {
          this.selectedShapeId = '';
      }
        this.updateTransformer();
      },
      updateTransformer() {
        const transformerNode = this.$refs.transformer.getStage();
        const stage = transformerNode.getStage();
        const {selectedShapeId} = this;

        const selectedNode = stage.findOne('#' + selectedShapeId);
        console.log(selectedNode.getClassName());
        // do nothing if selected node is already attached
        if (selectedNode === transformerNode.node()) {
          return;
        }

        if (selectedNode.getClassName() === 'Layer') {
          transformerNode.detach();
          stage.batchDraw();
          return;
        }

        if (selectedNode) {
```

```
          // attach to another node
          transformerNode.moveToTop();
          transformerNode.attachTo(selectedNode);
        } else {
          // remove transformer
          transformerNode.detach();
        }
        transformerNode.getLayer().batchDraw();
      },
async mounted() {
      this.changeRect();
      const id = this.$store.state.route.params.idb;
      const boardData = await BoardService.getBoardData(id);
      if (boardData.data.board == null) return;
      this.parseDataFromServer(boardData.data);
      this.createMenuList();
      this.addMenuButton();
    }
```

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

ІНТЕРАКТИВНА ВЕБ-ДОШКА ДЛЯ ЗБЕРІГАННЯ ОСОБИСТОГО МЕДІА-КОНТЕНТУ

Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проекту:

_____ Р.А. Гадиняк

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ Д.С. Дячук

2018

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Інтерактивна веб-дошка для збереження особистого медіа-контенту, яка являє собою веб-застосунок, створений на платформі Node.js з використанням фреймворку Vue.js.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1) функціональна працездатність компонентів веб-застосунку;

2) наявність доступу до медіа-файлів з хмарного сховища;

3) працездатність дошки при великій кількості даних;

4) забезпечення належного рівня безпеки даних;

5) зручність роботи з веб-застосунком;

6) відповідність дизайну вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Обрано декілька методик: модульне (тестування окремих компонентів веб-застосунку), інтеграційне (тестування компонентів, інтегрованих один з одним), системне (перевірка всієї системи на несправності).

Модульне тестування дозволяє перевірити коректність роботи окремих компонентів веб-застосунку, а саме:

- авторизацію та реєстрацію;
- зміну профіля;
- модифікацію дошки;
- інтерфейс користувача.

Завдяки інтеграційному тестуванню перевіряється правильність взаємодії між компонентами та зв’язків між ними. Зокрема, успішна передача даних від одного компонента до іншого при завантаженні нового медіа-файлу на дошку.

Системне тестування дозволяє перевірити правильність взаємодії СУБД та хмарного сховища з розробленим веб-застосунком.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується вручну. Перевірка працездатності веб-застосунку перевіряється шляхом:

1) Тестування коректної роботи завантаження медіа-файлів на дошку.
2) Тестування коректної роботи видалення медіа-файлів з дошки.
3) Тестування інтерактивних можливостей веб-застосунку.
4) Тестування продуктивності при великій кількості медіа-файлів на дошці.
5) Тестування правильності взаємодії з хмарним сховищем.
6) Статичного тестування коду;
7) Тестування інтерфейсу користувача.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ІНТЕРАКТИВНА ВЕБ-ДОШКА ДЛЯ ЗБЕРІГАННЯ ОСОБИСТОГО МЕДІА-КОНТЕНТУ

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проекту:

_____ Р.А. Гадиняк

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ Д.С. Дячук

2019

ЗМІСТ

1. Опис структури веб-застосунку

Веб-застосунок Інтерактивна дошка для зберігання медіа-контенту складається зі статичних веб-сторінок та веб-сторінок, вміст яких формується динамічно.

До статичних належать такі сторінки:

- Головна сторінка веб-застосунку.
- Сторінка реєстрації.
- Сторінка авторизації.

До динамічних належать наступні сторінки:

- Профіль користувача.
- Налаштування профілю користувача.
- Сторінка для перегляду та взаємодії з дошкою.
- Список публічних дошок.

Після авторизації кожна сторінка зверху має навігаційний бар з посиланнями на головну сторінку, профіль користувача, список публічних дошок та кнопку виходу з профілю.

2. Опис вмісту статичних веб-сторінок

Головна сторінка веб-застосунку (рис. 1) розповідає про його основне призначення та стисло описує головні функції. В кінці сторінки наведено приклад інтерактивної дошки, щоб користувач зміг наглядно побачити зразок результату, який він може отримати після роботи з дошкою.

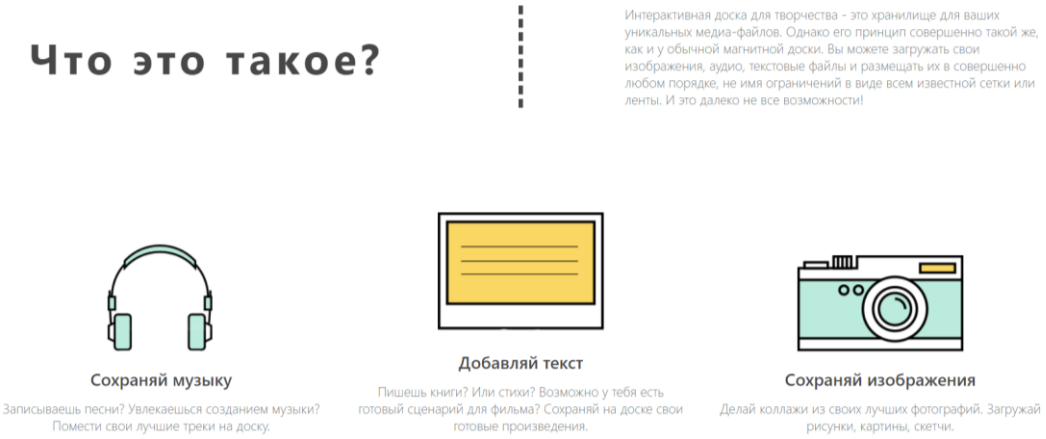


Рис. 1. Головна сторінка веб-застосунку. Опис та призначення

Сторінка реєстрації (рис. 2) містить спеціальну форму з полями для вводу персональних даних: унікального ім’я користувача, повного ім’я, електронної пошти та паролю. Користувач повинен заповнити усі необхідні поля та натиснути кнопку «Зареєструватися». Після успішної реєстрації його буде перенаправлено на його персональну сторінку.

Сторінка авторизації (рис. 3) містить форму для входу в аккаунт. Користувач повинен ввести своє унікальне ім’я та пароль. Після успішного входу в систему користувача буде перенаправлено на його персональну сторінку.

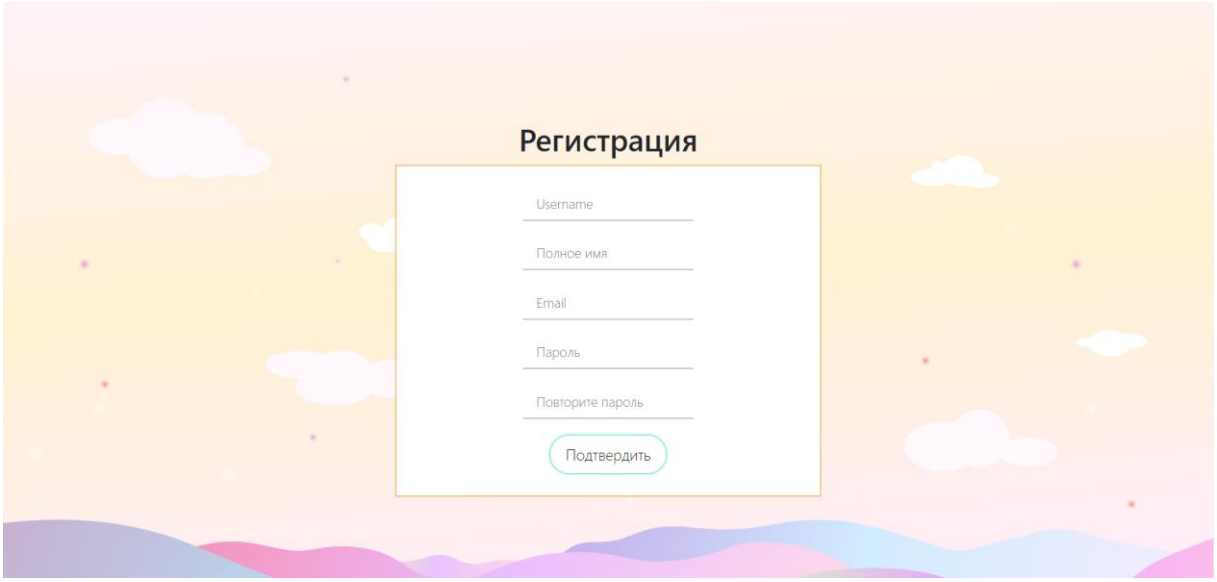


Рис. 2. Сторінка реєстрації

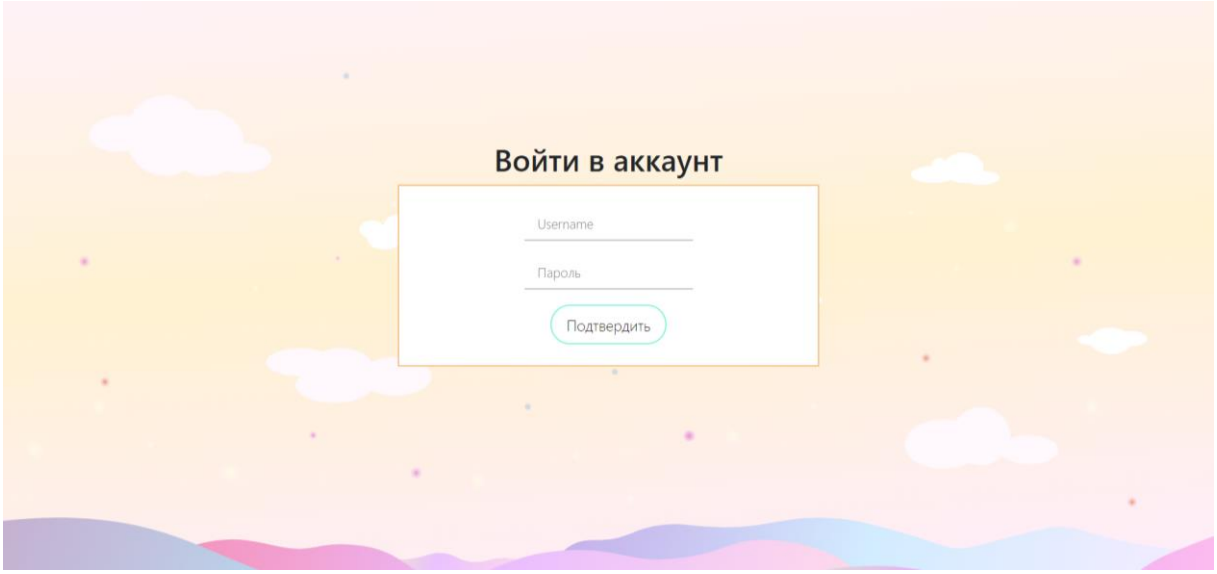


Рис. 3. Сторінка авторизації

3. Сторінка користувача

На сторінці користувача (рис. 4) показано основну інформацію про нього: унікальне ім’я, повне ім’я та опис профілю. За замовченням веб-застосунок налаштовує його аватар.

У розділі «Дошки» міститься список усіх дошок, які створив користувач. Усі дошки мають мітку публічна або приватна та іконку, за допомогою якої можна видалити обрану дошку.

На цій сторінці створюється нова дошка.

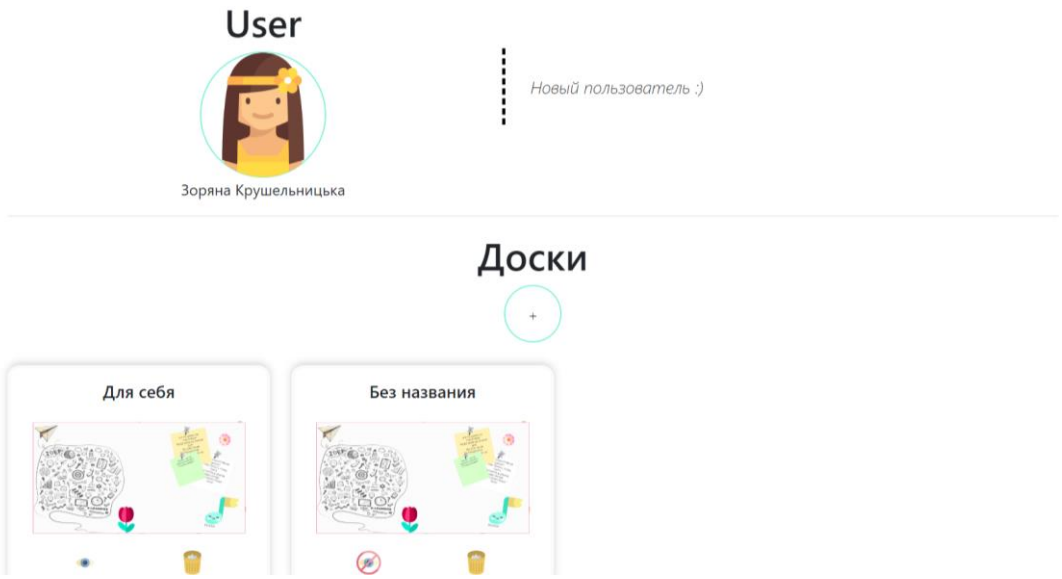


Рис. 4. Сторінка користувача

4. Взаємодія з дошкою

Дана сторінка (рис. 5) дозволяє переглянути власну дошку та модифікувати її.

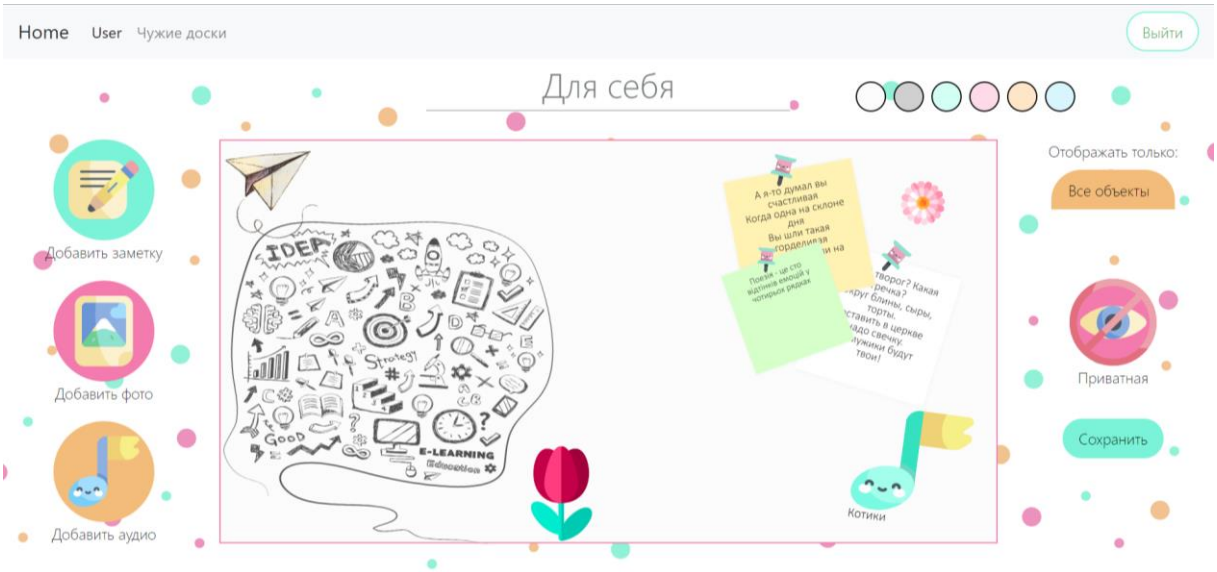


Рис. 5. Сторінка для взаємодії з веб-дошкою

Сама дошка займає майже весь простір сторінки. Зверху знаходиться текстове поле, яке одночасно показує назву дошки і дозволяє її змінити. Справа від нього розташований набір різнокольорових кнопок, за допомогою яких можна змінити колір дошки.

Зліва від дошки розміщено панель з трьох кнопок для додавання медіа-файлів: текстового нотатку, зображення та аудіо. Після натискання кожної з кнопок відкривається модальне вікно, у якому можна вибрати деталі оформлення медіа-файлу.

Щоб створити нотаток необхідно натиснути на кнопку «Додати нотаток», ввести текст у полі для вводу та обрати колір нотатка (рис. 6). Для завантаження об’єкта на дошку натиснути «Зберегти».

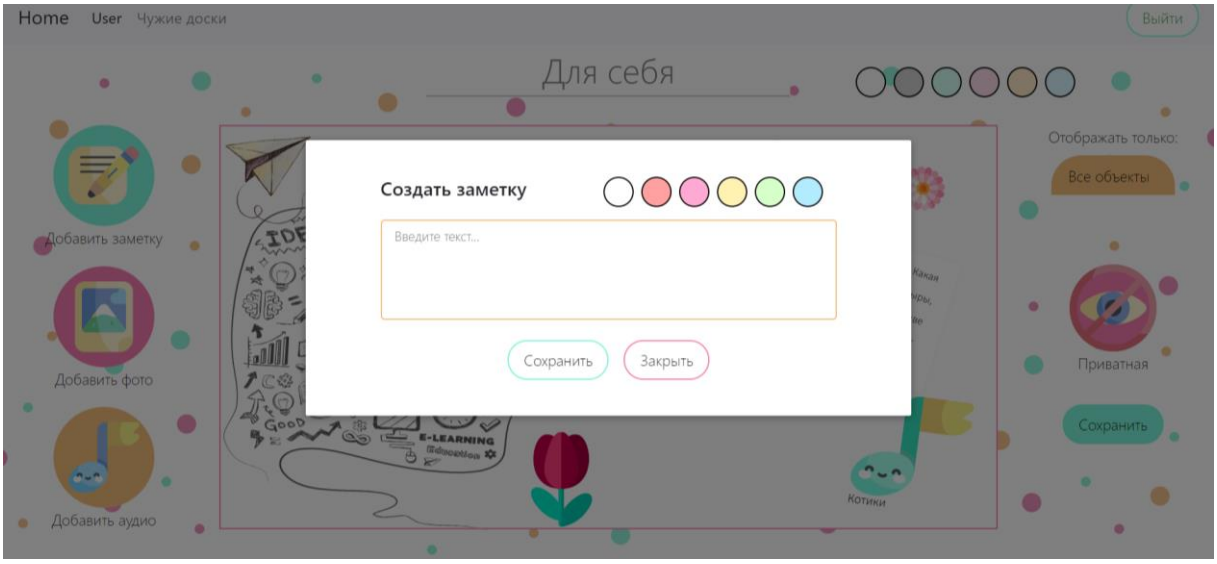


Рис. 6. Створення текстового нотатка

Щоб додати зображення на дошку необхідно натиснути на кнопку «Додати фото», у відкритому вікні натиснути «Обрати файл», а потім обрати оформлення зображення (рис. 7). Для завантаження медіа-файлу на дошку натиснути «Зберегти».

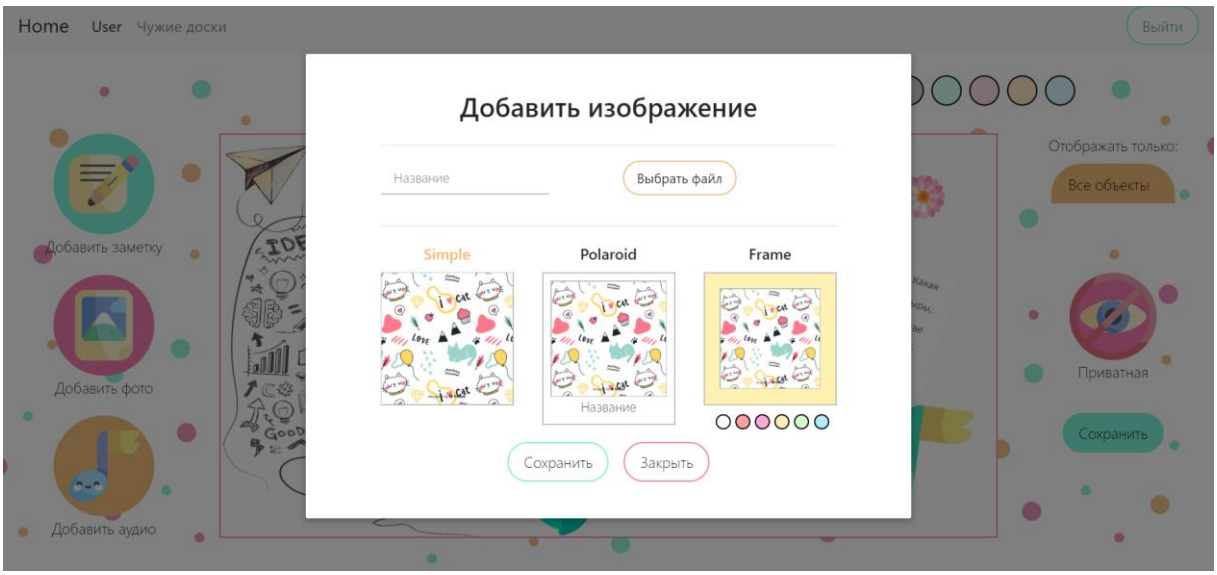


Рис. 7. Створення зображення

Щоб додати аудіо на дошку необхідно натиснути на кнопку «Додати аудіо», у відкритому вікні натиснути «Обрати файл», а потім обрати оформлення аудіо (рис. 8). Для завантаження медіа-файлу на дошку натиснути «Зберегти».

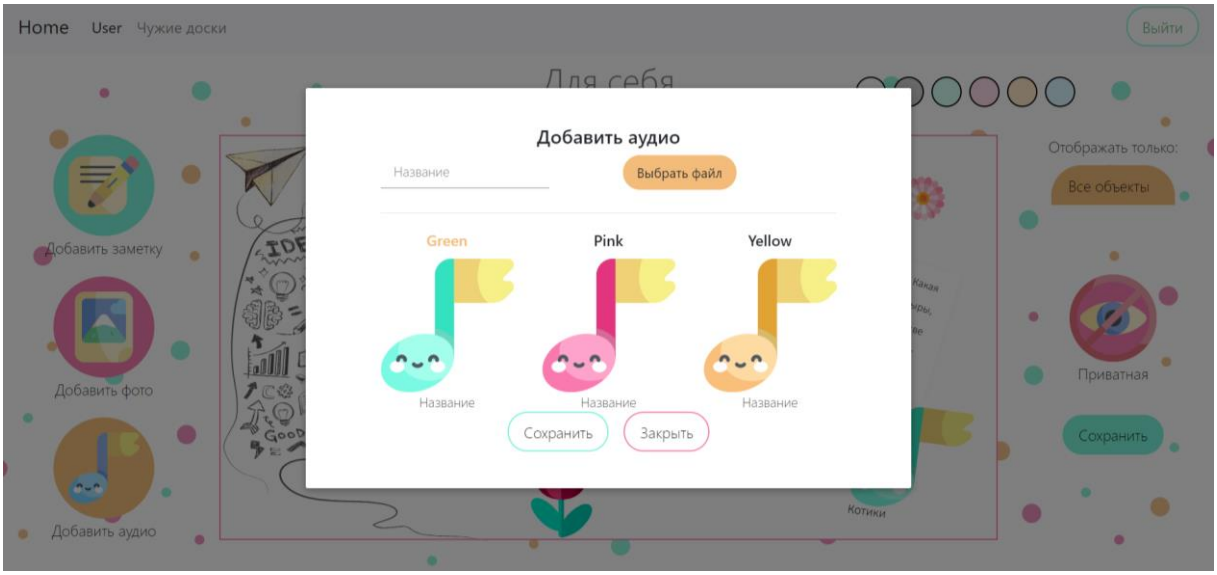


Рис. 8. Створення аудіо

Після завантаження медіа-файлів на дошку їх можна розміщати в будь-якому місці у межах дошки, змінювати їх розмір, крутити, компонувати один з одним. Аудіо-файли можна прослуховувати, натиснувши на них два рази.

Кожен об’єкт на дошці має меню (рис. 9) зі списком функцій, які можна виконувати над ним. Воно відкривається за натисканням миші, коли обрано певний медіа-файл та надає такий список функцій:

- Видалення об’єкта.
- Перетягування об’єкта на передній план.
- Перетягування об’єкта на задній план.
- Завантажити медіа-файл на комп’ютер.
- Зображення можна відкрити у повному розмірі.

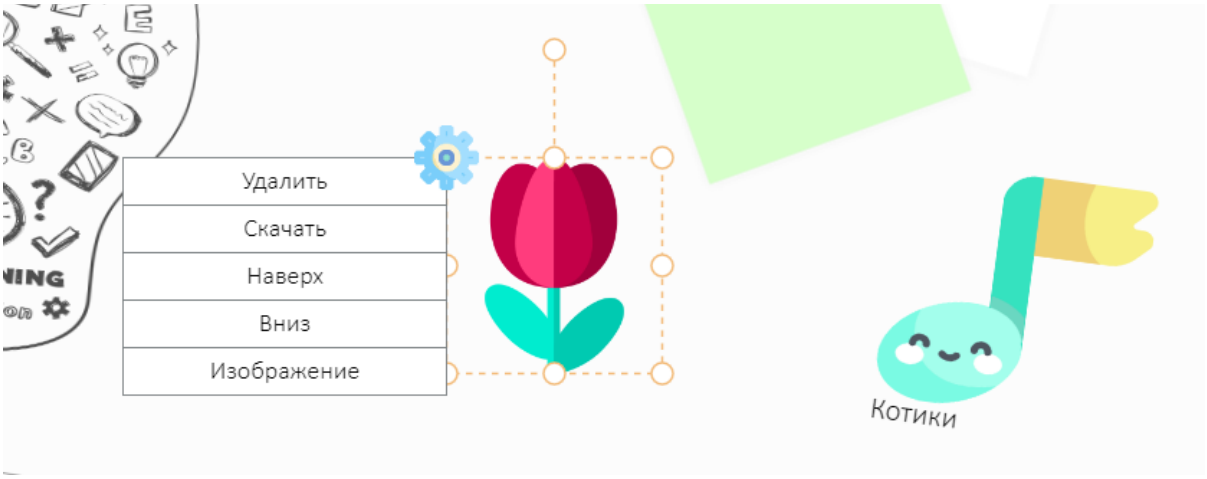


Рис. 9. Меню виділеного об’єкта

Справа від дошки знаходиться панель для фільтрації контенту. На дошці можуть бути показані медіа-файли усіх типів або якогось конкретного (тільки зображення, аудіо або текст). Знизу під панеллю є дві кнопки. Перша з них дозволяє змінити вид дошки: зробити її публічною або приватною. Друга кнопка зберігає стан дошки, зокрема місцезнаходження кожного об’єкта, його кут повороту, розмір тощо.

5. Зміна профілю користувача

Веб-застосунок надає можливість змінити параметри персонального профілю (рис. 10). За замовчуванням доступно шість видів аватарів. Можна змінити повне ім’я і опис профілю у відповідних текстових полях форми. Щоб

зберегти зміни необхідно натиснути «Зберегти». Кнопка «Назад» повертає на профіль користувача.

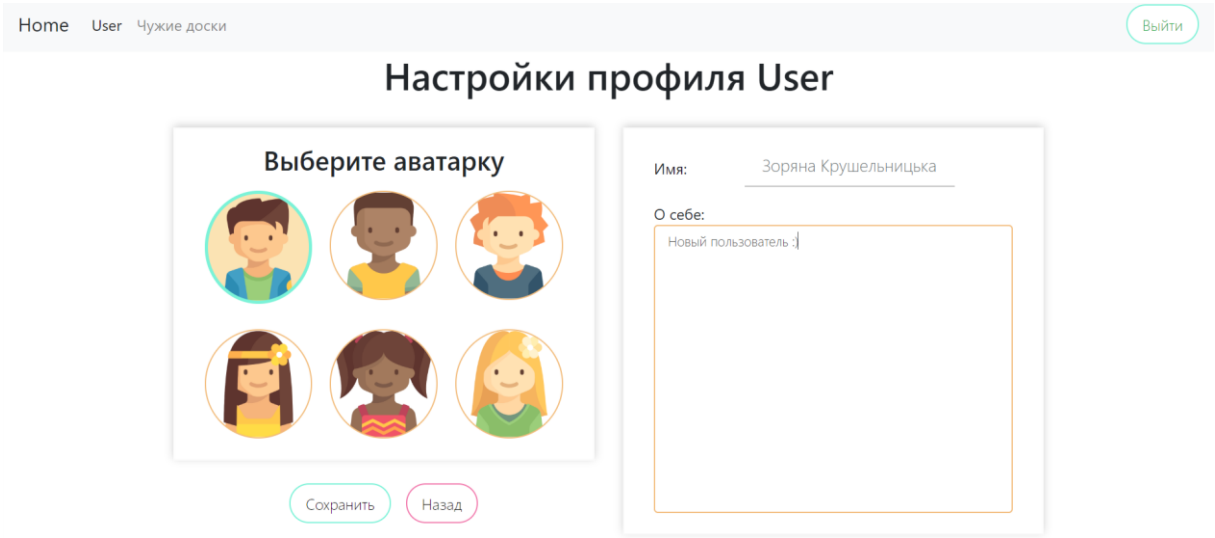


Рис. 10. Налаштування профілю користувача

6. Перегляд публічних дошок

На навігаційній панелі необхідно натиснути на посилання «Чужі дошки». Серед списку публічних дошок необхідно обрати одну, і веб-застосунок перенаправить відповідно на сторінку її перегляду (рис. 11). Вона має майже такий самий вигляд, як і сторінка зміни власної дошки. Відсутня ліва панель, панель зміни кольору дошку, кнопка «Зберегти», поле з назвою дошки недоступне для зміни. З’явилося поле з ім’ям автора дошки. У меню об’єкта доступна лише одна функція: переглянути зображення у повному розмірі.

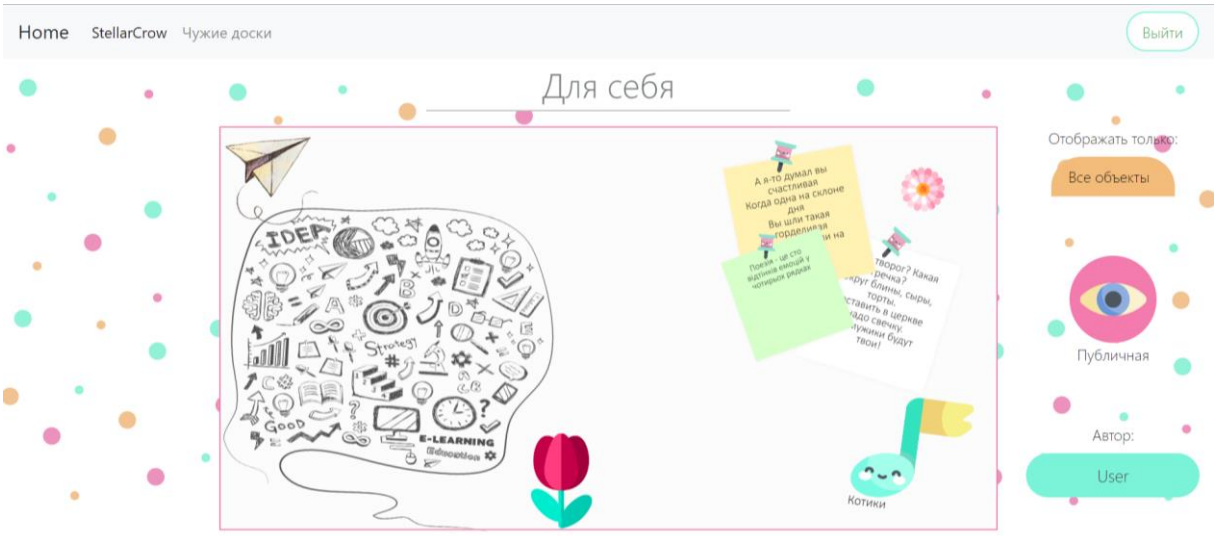


Рис. 11. Сторінка перегляду публічної дошки